UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER

INGENIERÍA DE TELECOMUNICACIÓN

Configuración de redes TSN síncronas para el transporte de network slices en 5G

Optimización del planificador mediante Machine Learning

Autor

Pablo Rodríguez Martín

Directores

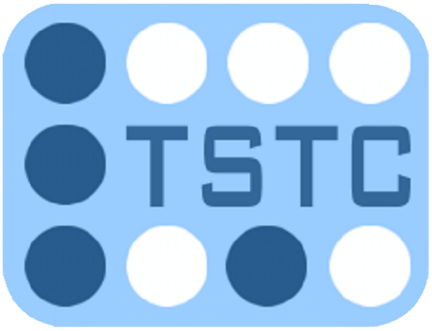
Pablo Ameigeiras Gutiérrez

Pablo Muñoz Luengo

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, septiembre de 2022



Configuración de redes TSN síncronas para el transporte de network slices en 5G

Optimización del planificador mediante Machine Learning

Autor
Pablo Rodríguez Martín

Directores
Pablo Ameigeiras Gutiérrez
Pablo Muñoz Luengo

Departamento de Teoría de la Señal, Telemática y Comunicaciones

—

Granada, septiembre de 2022

Configuración de redes TSN síncronas para el transporte de network slices en 5G: Optimización del planificador mediante Machine Learning

Pablo Rodríguez Martín

Palabras clave: Time-Sensitive Networking, Comunicaciones Síncronas, Time-Aware Shaper, IEEE, 3GPP, 5G, New Radio, Industria 4.0, IIoT, Network Slices, URLLC, Aprendizaje automático, Algoritmos Genéticos.

Resumen

En este documento se presenta el trabajo de investigación llevado a cabo para el proyecto 6G-CHRONOS de la Universidad de Granada y que tiene como objetivo la integración del conjunto de estándares del IEEE que conforman las redes deterministas *Time-Sensitive Networking* (TSN) junto con la quinta generación de tecnologías de redes móviles (5G) del 3GPP para la automatización de los procesos en la actual y creciente Industria 4.0. En concreto, este estudio se centra en el caso de TSN síncrono con la optimización del planificador *Time-Aware Shaper* (TAS), donde se han de tener en cuenta aspectos de especial relevancia como la sincronización de relojes entre los dispositivos, las fluctuaciones producidas por el procesamiento de las tramas en las NFV y el uso de *network slices* y colas de prioridad para la segmentación del tráfico con el fin de aislar y ajustar los tiempos de transmisiones periódicas de tráfico URLLC. Dichas comunicaciones poseen unos requisitos que vienen definidos principalmente por un retardo extremo a extremo y un *jitter* acotados que no pueden excederse, pues de lo contrario las aplicaciones críticas no serían realizables. Por otro lado, el uso de este tipo de redes de acceso inalámbricas ofrece movilidad para un mayor volumen de dispositivos conectados y con un menor coste de despliegue asociado cuando se alquila la infraestructura de un operador, aunque también implican un mayor retardo al hacer uso del canal radio para la transmisión y recepción de la información. Por ello es importante planificar unos tiempos que se adecuen a los requerimientos de cada comunicación, con lo que se ha desarrollado un modelo de TAS en Python orientado a la minimización del retardo de los distintos flujos a través de las posibles rutas. De este modo, se ha optado por un esquema *zero-wait*, el cual evita el retardo ocasionado por la espera en colas y que se añade a los retardos de conmutación, transmisión y propagación; aunque esto provoca la aparición de intervalos de tiempo en los que no es posible la transmisión de ninguna de las otras colas de prioridad dado su tamaño insuficiente, disminuyendo así el *throughput* en los enlaces. Es por este motivo por el que se ha decidido tener en cuenta estos huecos para evaluar distintas combinaciones de rutas para cada flujo y con ello maximizar también la utilización de los enlaces a través de métodos de Inteligencia Artificial, más concretamente con los algoritmos genéticos y el aprendizaje automático supervisado en base a sus resultados. De esta forma, se ha concluido con un modelo efectivo capaz de optimizar a través de pesos los resultados de ambos parámetros según la topología de red desplegada y las características de los flujos. Además, ofrece resiliencia ya que se analizan a priori fallos en la red para mantener en medida de lo posible el rendimiento logrado a través de la reconfiguración de los tiempos en rutas alternativas. Finalmente, se ha adaptado el modelo a las peculiaridades de 5G a través de bandas de guarda para la absorción de las fluctuaciones con los *bridges* 5G como caja negra, viendo así el gran problema que suponen los retardos entre sus puertos virtuales.

Configuration of TSN networks for the transport of 5G network slices: Optimization of the scheduler through Machine Learning

Pablo Rodríguez Martín

Keywords: Time-Sensitive Networking, Synchronous Communications, Time-Aware Shaper, IEEE, 3GPP, 5G, New Radio, Industry 4.0, IIoT, Network Slices, URLLC, Machine Learning, Genetic Algorithms.

Abstract

This document presents the research work carried out for the 6G-CHRONOS project of the University of Granada, which aims at integrating the set of IEEE standards that make up the deterministic Time-Sensitive Networking (TSN) together with the 3GPP's fifth generation of mobile network technologies (5G) for the automation of processes in the current and growing Industry 4.0. In particular, this study focuses on the case of synchronous TSN with the optimisation of the Time-Aware Shaper (TAS), where aspects of special relevance such as clock synchronisation between devices, jitter caused by frame processing in NFVs and the use of network slices and priority queues for traffic segmentation in order to isolate and adjust the timing of periodic transmissions of URLLC traffic must be taken into account. Such communications have requirements that are mainly defined by a bounded end-to-end delay and jitter that cannot be exceeded, otherwise critical applications would not be feasible. Furthermore, the use of this type of wireless access networks offers mobility for a higher volume of connected devices and with a lower deployment cost when infrastructure is rented from an operator, although they also imply a higher delay when making use of the radio channel for the transmission and reception of information. For this reason, it is important to plan times that adapt to the requirements of each communication, so a TAS model has been developed in Python aimed at minimising the delay of the different flows through the possible routes. In this way, a zero-wait scheme has been chosen, which avoids the delay caused by waiting in queues and which is added to the switching, transmission and propagation delays; although this causes the appearance of time intervals in which the transmission of any of the other priority queues is not possible due to their insufficient size, thus reducing the throughput in the links. Consequently, these gaps have been taken into account in order to evaluate different combinations of routes for every flow and thus also maximise the utilisation of the links through Artificial Intelligence methods, more specifically with genetic algorithms and supervised machine learning based on their results. In this way, we have concluded with an effective model capable of optimising through weights the results of both parameters according to the network topology deployed and the characteristics of the flows. Moreover, it offers resilience since network failures are analysed a priori in order to maintain as much as possible the performance achieved through the reconfiguration of the times on alternative routes. Finally, the model has been adapted to the peculiarities of 5G through guardbands for jitter absorption with 5G bridges as a black box, thus seeing the major problem of delays between its virtual ports.

Yo, **Pablo Rodríguez Martín**, alumno de la titulación Máster en Ingeniería de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75571979-M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Pablo Rodríguez Martín

Granada a 9 de septiembre de 2022.

D. **Pablo Ameigeiras Gutiérrez** y D. **Pablo Muñoz Luengo**, Profesores del Área de Ingeniería Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Configuración de redes TSN síncronas para el transporte de network slices en 5G, optimización del planificador mediante Machine Learning***, ha sido realizado bajo su supervisión por **Pablo Rodríguez Martín**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 9 de septiembre de 2022.

Los directores:

Pablo Ameigeiras Gutiérrez **Pablo Muñoz Luengo**

Agradecimientos

Quiero dar las gracias en primer lugar a todas las personas que han estado presentes a lo largo de las diferentes etapas de mi formación, en especial a todos/as aquellos/as profesores/as y compañeros/as por haberse esforzado en todo momento a ayudarme y haberme enseñado los conocimientos y valores con los que finalmente realizo este proyecto.

En segundo lugar, quiero expresar mi gratitud a mis tutores, Pablo A. y Pablo M., por haberme acompañado y guiado en el desarrollo de este proyecto en el que he logrado aprender mucho sobre los ámbitos que más me han motivado su estudio durante la carrera. También reconocer el gran apoyo recibido por parte de los/as compañeros/as del grupo de investigación WiMuNet, pues se han mostrado dispuestos a ofrecer en todo momento su ayuda y a proveerme del material necesario.

Por último, quiero mostrar mi especial agradecimiento a mi familia y amigos/as por el apoyo incondicional recibido, pues siempre han estado a mi lado durante mis éxitos y fracasos, me han dado ánimos para proseguir y nunca han dudado de que lograría alcanzar mis metas, sobre todo en los momentos más difíciles. Soy quien soy gracias a ellos/as.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La llegada de 5G ha supuesto una auténtica revolución en el mundo de las telecomunicaciones, principalmente para la Industria 4.0[1] al permitir compartir información entre máquinas a través de sensores y controladores de gran heterogeneidad conectados a una misma red privada de acceso inalámbrico, lo que en la actualidad se conoce como *Industrial Internet of Things (IIoT)*. Esto evita el alto coste que supone todo un despliegue de cableado a la vez que proporciona una mayor movilidad de los dispositivos. Además, se realiza de una forma más estandarizada con *Open Radio Access Network (O-RAN)*, con lo que se hace posible la interoperabilidad entre distintas tecnologías y fabricantes de mayor o menor complejidad. El principal objetivo de todo ello es conectar un gran volumen de dispositivos que se hacen cargo de la automatización de los procesos industriales con el fin de lograr una salida de los productos al mercado mucho más segura, económica y eficiente con el uso de técnicas más avanzadas como pueden ser aquellas basadas en la nube e incluso en Inteligencia Artificial. Es por este motivo por el que un gran número de empresas y operadores de red están trabajando ya en el despliegue de esta infraestructura. Se prevé que este nuevo modelo genere más de $138M solamente en facturación por parte de los operadores móviles y que para el año 2030 el empleo de entre el 3 % y el 14 % de la población mundial se vea afectada por esta nueva industria, migrando desde perfiles técnicos especializados hacia otros más enfocados en el sector tecnológicos.

Sin embargo, el mayor problema del uso de comunicaciones inalámbricas es que conlleva un mayor retardo junto con la aparición de lo que se denomina *jitter*, es decir, la variación del retardo de los paquetes que un dispositivo recibe en función de las condiciones del canal radio, pues el sistema planificador de la estación base de 5G o *gNB* asignará los recursos (ancho de banda, modulación, razón de código, etc.) de una forma u otra en función de éstas; además del propio tráfico que soporta la red de transporte en un determinado instante, llegando así a incrementarse la latencia en la llegada de paquetes a destino cuanto mayor sea éste y que puede perjudicar a los requisitos extremo a extremo en función del tipo de encaminamiento configurado y la carga para la que esté diseñada dicha red. Además, es importante tener también en cuenta otros aspectos como la distancia entre los dispositivos y el *gNB* o el tamaño de los paquetes, dado que el *throughput* puede verse más limitado e incrementar dicho retardo. Esto en principio no afecta en exceso al tráfico más genérico ya que suele ser relativamente bajo en las redes actuales con los mecanismos que poseen para evitar congestión, pero es realmente crítico para aplicaciones de tiempo real que requieren hacer uso de comunicaciones con KPIs en el *slice* 5G de *Ultra-Reliable Low-Latency Communications (URLLC)*[2] ya que en ellas se persigue el mínimo retardo

posible en la red por debajo de un límite superior (e.g. 1 ms) con una fiabilidad de que al menos el 99.999 % de los paquetes llega a su destino. Entre este tipo de aplicaciones se pueden encontrar la conducción autónoma de vehículos, sistemas de emergencia, cirugías en remoto, automatización de maquinaria robotizada, aplicaciones de realidad aumentada, logística, etc.

1.1. Motivación y objetivos del proyecto

Es por todo ello por lo que desde la Release 16 del *3rd Generation Partnership Project (3GPP)*[3] se propone el uso del estándar de *Time-Sensitive Networking (TSN)*[4] como integración con el sistema de segmentación virtual del tráfico de *network slicing* 5G con el fin de obtener una red determinista. En este tipo de redes se busca la sincronización entre elementos para conceder una mayor prioridad a las transmisiones de este tipo de aplicaciones críticas de la Industria 4.0 de modo que la distribución de probabilidad del retardo quede por debajo de ese límite con tal fiabilidad. Dicha integración puede verse con el esquema de la Figura 1.1, donde la red TSN aparece gestionada por un mismo controlador tanto en la conexión del transmisor y el receptor en que el sistema de 5G actúa como un elemento más de dicha red; como dentro de la red de transporte de 5G, donde se interconectará con el resto del sistema a través de las funciones de red virtualizadas o *Network Function Virtualization (NFV)*, las cuales se distribuyen para dotar al sistema de una mayor flexibilidad.

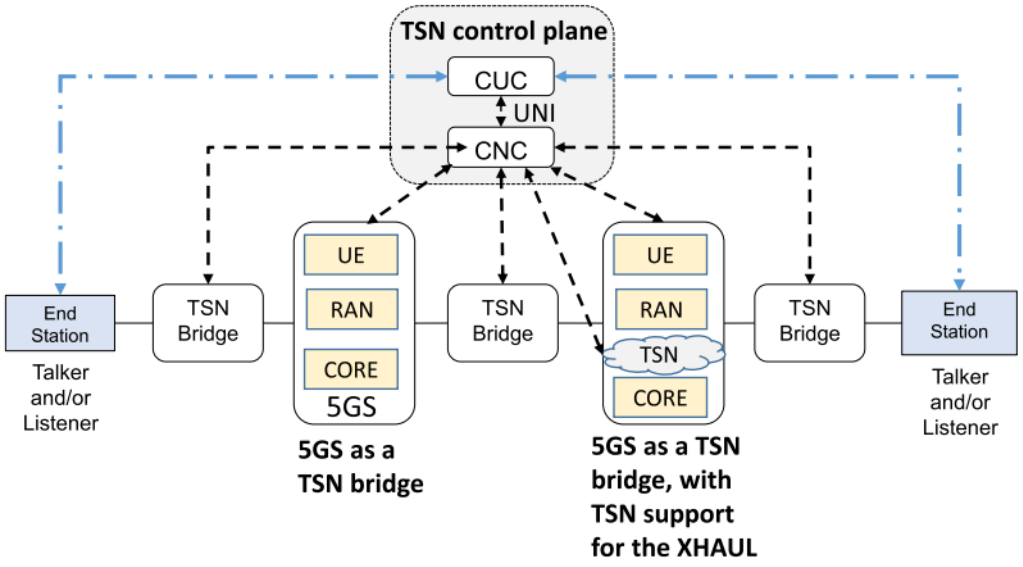


Figura 1.1: Esquema de integración de O-RAN 5G con red determinista TSN [4]

TSN actúa por tanto de forma análoga a *Software-Defined Network (SDN)*, donde aparece un controlador centralizado que se conoce como *Centralized Network Controller (CNC)* y que con aplicaciones de más alto nivel es capaz de establecer políticas de seguridad (e.g. ACLs), particionar la red física en diferentes *Virtual Local Area Network (VLAN)*, configurar topologías los protocolos de encaminamiento de cada flujo o gestionar los recursos hardware de cada equipo; pero también intervenir en el análisis de los

requisitos de usuario los para la posterior configuración de los tiempos de transmisión asignados a diferentes clases de tráfico según su prioridad. Con ello, tenemos un plano de datos por el que se encaminan las tramas hacia su destino y un plano de control con el que se configuran estos equipos.

Utilizar TSN también dentro del bloque de red de transporte o *midhaul* permite reducir todavía más el retardo dado que las unidades de procesamiento de la señal se distribuyen a lo largo de una red determinista para lograr una mayor escalabilidad dentro del segmento de 5G. Entre estas NFV se encuentran el *Radio Unit (RU)*, el cual se encarga de muestrear, filtrar y convertir (A/D o D/A) la señal en fase y cuadratura; el *Distributed Unit (DU)*, que realiza el corte o *split* dentro de la primera capa física del modelo OSI, lo que se define a diferentes niveles en *evolved Common Public Radio Interface (eCPRI)*; y el *Centralized Unit (CU)*, el cual ya maneja tramas de la capa MAC en adelante para los planos de datos y de control. El DU es el que más retardo introduce en función de la opción donde se produce tal división, por lo que se hace importante el uso de TSN en este segmento con el fin de acotar estas latencias. Además, al tratarse de funciones virtualizadas, dicha integración se enfrenta al problema de las fluctuaciones temporales que se producen debido a la gestión de interrupciones del sistema operativo que coordina los recursos hardware del servidor, por lo que esto podría llevar a importantes desajustes temporales en las transmisiones, lo cual es crítico en TSN síncrono. En la Figura 1.2 puede verse esta distribución de NFVs junto a sus respectivas ubicaciones en una red de radioacceso y transporte hasta el núcleo de la red 5G.

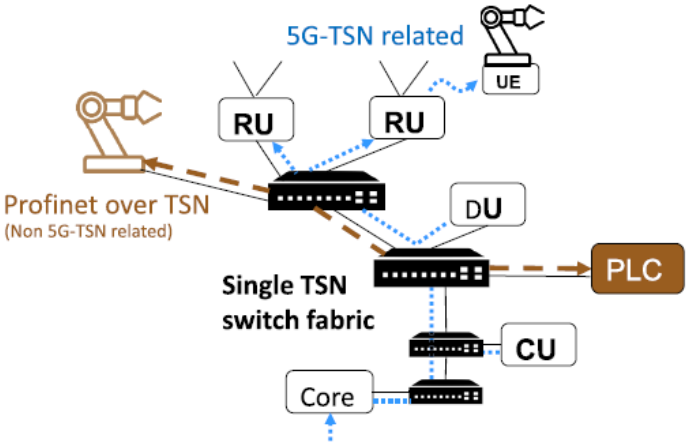


Figura 1.2: Red de transporte Xhaul en 5G [4]

Gracias a ello, dicho segmento puede dividirse también virtualmente en función de lo que se denomina *slice* y con el que se comunican a través de diferentes sesiones los dispositivos en 5G, pudiendo asignar tanto los recursos radio como los nodos de la red de transporte en función de la relevancia o prioridad de cada clase de tráfico, en este caso siendo el de URLLC el de mayor interés dada su criticidad en tiempo. Por ejemplo, para un tipo de tráfico diferente a éste y de menor prioridad como podría ser la comunicación móvil *enhanced Mobile BroadBand (eMBB)*, la ruta seguida por las tramas puede diferir en cuanto a nodos por los que pasa desde que una estación base la recibe hasta que llega a la red troncal de 5G y viceversa. Como se ve en la Figura 1.3, resulta necesario que además exista una sincronización entre los diferentes nodos TSN, pero también dentro del propio sistema de 5G dado que se ha de tener en cuenta el retardo en la asignación de dichos recursos, en ambos casos a través de *generic Precision Time Protocol (gPTP)*.

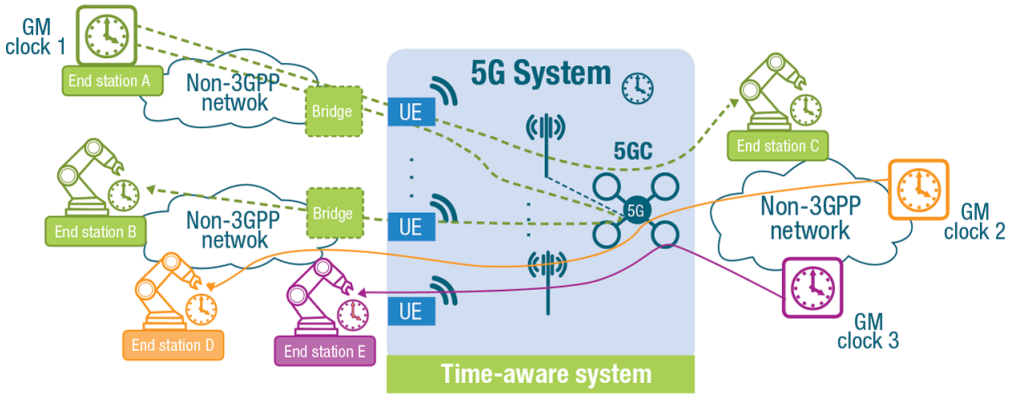


Figura 1.3: Esquema de Bridge 5G en la Industria 4.0 [5]

Sin embargo, el principal problema de la configuración actual es que la asignación en TSN de estos recursos temporales se hace en función de la demanda según algoritmos lineales de optimización como *Satisfiability Modulo Theories (SMT)* o *Mixed-Integer Linear Programming (MILP)* como muestran F. Pozo et al. en [6], algo que parece no ser suficiente ya que al tratarse de un problema NP-duro con un espacio de búsqueda exponencial a medida que aumenta el volumen de tráfico soportado por este tipo de redes determinista y a mayor complejidad de la topología, mayor será el tiempo de cómputo necesario para converger en una solución adecuada. En cambio, existen modelos que permiten una mejor optimización, como es el caso de planificación basada en colonias de hormigas que propone Y. Wang et al. [7], aunque no son aplicables en situaciones muy cambiantes en las que se requiere de una rápida adaptación a las circunstancias, como puede ser por ejemplo el cambio de la topología debido al fallo de uno de los nodos, por lo que esto puede desencadenar en problemas críticos; además de que no miden la eficiencia en el uso de recursos. Por ello, cuanto mayor sea la granularidad de los tiempos asignados a cada clase de tráfico, más complejo será el problema y por ello más tiempo llevará su cómputo.

Tal es la importancia de las redes deterministas en la nueva industria que han aparecido alianzas como *5G Alliance for Connected Industries and Automation (5G-ACIA)*[5] en la que participan empresas como Intel, Microsoft, Huawei, Nokia, Telefónica o Cisco con el fin de perseguir conjuntamente estos objetivos, proponiendo el caso de TSN síncrono como la mejor opción para reducir lo máximo posible estos retardos de forma que pueda acelerar los procesos de respuesta en este sector y segmentar los diferentes flujos de tráfico en *slices* y en función de su prioridad. Con esto, el proyecto 6G-CHRONOS se ha constituido en el seno del grupo de investigación WiMuNet de la Universidad de Granada con el fin de posibilitar un *framework* capaz de dar solución a una integración determinista y de bajo retardo para redes móviles industriales 5G. Nuestro estudio se pretende enfocar por tanto en el caso de TSN síncrono para su integración con *network slices* de 5G, el cual hace uso de esta sincronización entre bloques para establecer los tiempos de los puertos en los que se transmite cada trama en función de su prioridad, principalmente de aquellas destinadas a comunicaciones de aplicaciones críticas. La intención de ello se extiende al uso algoritmos de Inteligencia Artificial, concretamente de *Machine Learning*, con el fin de hallar de forma *offline* cuáles son los intervalos y periodos de tiempo óptimos asignados a cada una de estas clases para que el CNC aprenda y sepa en todo momento inferir en una configuración u otra según la situación para cada nodo de la red con el objetivo de

reducir lo máximo posible este retardo y responder ante fallos. Para ello, una vez conocido el ámbito del proyecto, se puede proceder a enumerar los objetivos que han surgido en su planteamiento y que se persiguen a lo largo de éste:

- **Análisis del impacto de 5G en la Industria 4.0 y requisitos de tiempo real**. Recopilación de posibles aplicaciones críticas que pueden verse beneficiadas con esta integración de ambas tecnologías en el ámbito industrial. Necesidad de comunicaciones inalámbricas en los procesos de automatización de IIoT a través de los tipos de comunicaciones definidos en función de las restricciones que imponen sus aplicaciones. Técnicas de despliegue de redes industriales y mercado de soluciones, evolución de redes inalámbricas y Ethernet.

- **Estudio acerca del funcionamiento de TSN**. Estandarización a día de hoy del IEEE 802.1Q, más concretamente para el caso de TSN síncrono; y medidores de desempeño en los que se mueven los protocolos. Posibles ámbitos de aplicación y priorización del tráfico más crítico sobre el resto. Control y gestión de la red, tipos de gestores TSN existentes y configuración de los tiempos en los nodos. Acceso al medio físico y sincronización de equipos de red para el caso del modelo síncrono. Uso de TSN en la Industria 4.0. Análisis de los tipos de planificadores.

- **Análisis sobre la actual tecnología 5G** y estudio de las posibles fuentes de retardo y *jitter* por las que se pueden ver afectadas las aplicaciones de tiempo real. Estructura de control y gestión de los nodos así como el plano de datos. Definición de los elementos que componen su arquitectura y metodología de comunicación con la red TSN. Segmentación virtual de los recursos de la red a través de *network slices* y tipos de comunicaciones que habilita mediante las funciones virtualizadas. Fundamentos teóricos y técnicos sobre su funcionamiento, diferencias respecto a las anteriores tecnologías de redes móviles.

- **Investigación sobre el estado del arte en la integración entre 5G y TSN**. Estudio acerca de la sincronización entre relojes de ambos sistemas considerando el segmento de TSN dentro del propio *bridge* 5G. Comunicación entre módulos para la configuración de rutas y tiempos y funcionamiento de las sesiones por *slice*. Estructura y funcionamiento de la red de transporte actual 5G y adaptación a comunicaciones TSN. Solución de la problemática del indeterminismo con los elementos virtualizados y el segmento de radioacceso. Comunicación puente de formatos entre ambas tecnologías, módulos que intervienen y procedimientos. Rendimiento de los planificadores TSN en esta integración.

- **Análisis sobre las diferentes técnicas de Inteligencia Artificial**. Estudio sobre la viabilidad del uso de técnicas de *Machine Learning* en las redes TSN-5G. Selección de la más adecuada para el modelo de planificación en el caso síncrono.

- **Diseño y desarrollo de la integración TSN y 5G**. Implementación de un modelo que haga uso de *Machine Learning* para la optimización de la configuración de la red TSN junto con 5G en tal escenario a través de casos experimentales *offline*.

- **Evaluación de eficiencia y viabilidad** del sistema en cuanto a retardo y complejidad. Estudio de herramientas que expliquen el comportamiento de las variables del modelo aprendido sobre el retardo.

Para todo ello se aplicarán los conocimientos adquiridos a lo largo de los diferentes cursos del Máster de Ingeniería de Telecomunicación en la Universidad de Granada, y más concretamente en la rama de telemática, para encontrar la solución que mejor convenga dadas las circunstancias que propician el entorno de la investigación.

1.2. Planificación y metodología del proyecto

Dado que este proyecto se basa principalmente en la investigación y el posterior desarrollo de un modelo experimental, se han definido diferentes fases para llevarlo a cabo.

1. **Fase de investigación.** Se lleva a cabo un proceso de aprehensión de conocimientos relacionados con los diferentes ámbitos del proyecto con la búsqueda de información a través del temario estudiado a lo largo de la carrera, documentos bibliográficos y páginas de referencia en Internet como es el caso de publicaciones científicas relacionadas con este tipo de tecnologías, principalmente acerca de la Industria 4.0, las redes TSN síncronas, las redes móviles 5G y la integración de todo ello, así como de las posibles técnicas de Inteligencia Artificial que puedan llevar a una configuración optimizada. Estos conocimientos se anotan para las siguientes fases del proyecto.

2. **Fase de preparación.** Se estudian diferentes posibilidades acerca de su implementación basadas en las conclusiones y requisitos obtenidos a partir de las ideas desarrolladas con la fase de investigación. Se tratarán así las diferentes problemáticas que puedan encontrarse con el fin de hallar una solución real y práctica para esta integración. También se organizarán en base a esto las herramientas necesarias para el desarrollo del proyecto en fases posteriores.

3. **Fase de desarrollo y experimentación.** A partir de los conocimientos adquiridos con la investigación y el material disponible se llevará a cabo el desarrollo de un modelo experimental de planificador TSN para una topología real con la extracción del retardo *end-to-end* de las tramas para cada flujo a través de su identificador y la utilización de los enlaces como parámetros de acuerdo a los valores establecidos como de interés. Con ello se seguirá desarrollando para su adaptación al caso de 5G.

4. **Fase de resultados.** Se reunirán los resultados obtenidos de esta experimentación para formar una base sólida en el desarrollo del sistema. Se diseñará un algoritmo simplificado de optimización de modo que sea posible obtener para cada situación de la red la mejor configuración posible a través de un método de *Machine Learning* capaz de aprender estos resultados para luego ofrecer una salida que se adapte de forma más dinámica según estas entradas.

5. **Fase de pruebas.** Se realizarán las pruebas convenientes y posibles correcciones sobre el modelo con el fin de ajustar los parámetros que lo asemejen a la realidad en base a los resultados.

6. **Fase de redacción de la memoria.** Se reunirán todos los aspectos del proyecto para reflejarlos de forma escrita en una única memoria que dé a conocer su proceso y las conclusiones obtenidas en su realización.

Si se distribuyen estas fases de una forma más concisa a lo largo del tiempo se podría modelar la elaboración del proyecto mediante el siguiente diagrama de Gantt:

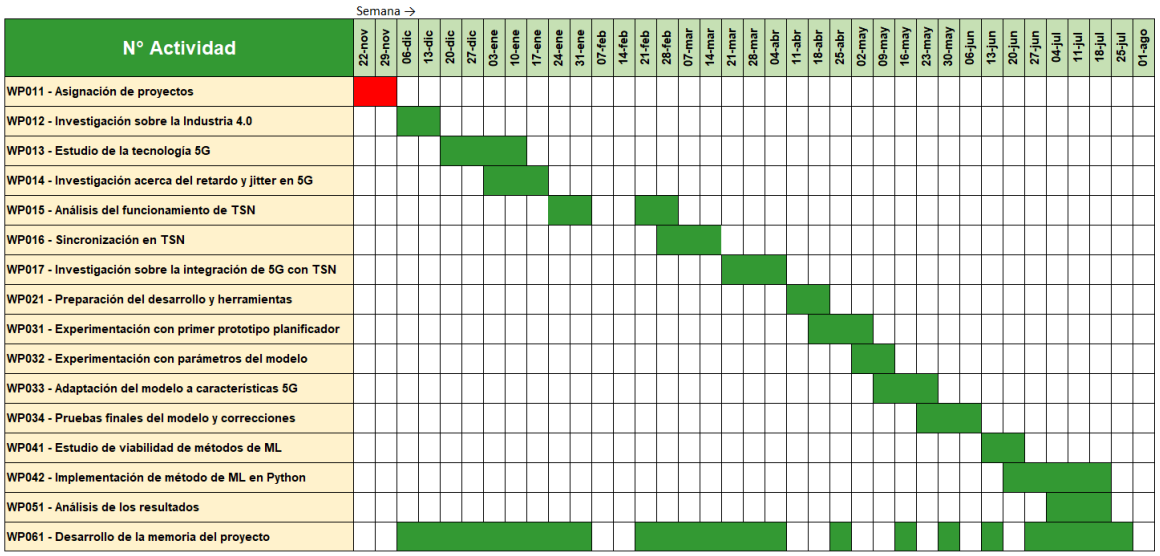


Figura 1.4: Diagrama de Gantt sobre la planificación del proyecto

1.3. Presupuesto del proyecto

Para comenzar con este proyecto se ha elaborado a priori un presupuesto general con la Tabla 1.1, en la que se recoge el gasto total de los diferentes elementos que lo componen, así como el coste asociado al tiempo invertido en la investigación y desarrollo del sistema como el beneficio esperado de ello al aplicar una tarifa por el despliegue de la red. Al tratarse de un proyecto de ingeniería en un ámbito más académico, se ha establecido un precio de 12€/hora. En calidad de investigador del grupo WiMuNet, el acceso al supercomputador de la UGR es gratuito.

Detalles	Cantidad	Precio por unidad
Materiales (-)		
Equipo computador para desarrollo	1x	850,00€
Python y librerías	1x	0,00€
Acceso Supercomputador UGR	1x	0,00€
Sub-total		850,00€
Tiempo empleado (+)		
Fase de investigación	17x 20h/semana	4.080,00€
Fase de preparación	2x 5h/semana	120,00€
Fase de desarrollo y experimentación	12x 15h/semana	2.160,00€
Fase de resultados y pruebas	6x 10h/semana	720,00€
Fase de redacción de la memoria	25x 5h/semana	1.500,00€
Tutorías docentes	25x 3h/semana	900,00€
Tiempo total empleado	790 horas	9.480,00€
Despliegue de la red. Operación (+)		
Configuración por nodo TSN + controlador	6+1x	700,00€
Configuración por nodo 5G	1x	500,00€
Sub-total		1.200,00€
Coste total del proyecto (sin I.V.A.)		11.530,00€
Coste total del proyecto (21 % I.V.A.)		13.951,3€

Tabla 1.1: Presupuesto general confeccionado previamente al proyecto

1.4. Estructura de la memoria

El presente documento se divide en seis capítulos, los cuales se detallan a continuación:

- **Capítulo 1. Introducción.** Breve descripción introductoria a la motivación que insta a la realización de este proyecto y definición de los objetivos principales que se pretenden lograr, así como la planificación temporal, el presupuesto del proyecto y la estructuración del contenido llevada a cabo.

- **Capítulo 2. Estado del Arte.** Investigación documental para trascender en el conocimiento del objeto de estudio, adquiriendo nuevas competencias que permitan realizar un análisis crítico para la toma de decisiones. Asimismo, se centra en la caracterización de la integración de sistemas 5G con TSN síncrono estudiando primero ambas partes por separado para conocer sus limitaciones, principalmente en cuanto a retardo y *jitter*; no sin antes realizar un estudio sobre el contexto de la Industria 4.0 que acontece por el momento y las ventajas del uso de estas tecnologías en aplicaciones críticas que requieren muy bajo retardo con una alta fiabilidad. Con ello, se analiza la bibliografía ya existente acerca de este tipo de integración y las posibles variables de entrada que pueden ser de interés durante el desarrollo.

- **Capítulo 3. Definición del problema y soluciones propuestas.** Se definen y explican las diferentes problemáticas encontradas en el estudio del estado del arte sobre la integración de ambas tecnologías con el fin de desarrollar soluciones que optimicen la planificación de TSN síncrono en su integración con el *bridge* 5G. Se divide por tanto en la resolución de un planificador optimizado para TSN síncrono y en su adaptación al caso de las fluctuaciones ocasionadas por las NFV.

- **Capítulo 4. Herramientas software y hardware orientadas al modelo.** Se definen y explican las diferentes herramientas hardware y software que se emplearán en el diseño y el desarrollo del proyecto, contando así con los recursos utilizados en el modelo a partir de la técnica de aprendizaje automático más conveniente.

- **Capítulo 5. Diseño del sistema.** Se procede a implementar el sistema en base a los conocimientos aprehendidos durante la investigación. Para ello se crea un modelo de aprendizake automático que dé solución a la planificación temporal de los recursos de la red TSN operando en conjunto con la red de acceso y transporte 5G como caja negra. Con ello se realizará un análisis de los resultados para comprobar el funcionamiento del sistema en su conjunto.

- **Capítulo 6. Conclusiones y trabajo futuro.** Se llevan a cabo las conclusiones del proyecto a través de los resultados obtenidos y se propone una línea de trabajo futuro basado en las conclusiones para la mejora del sistema en el futuro a corto y medio plazo.

Capítulo 2

Estado del Arte

2.1. La Industria 4.0

La historia de la humanidad ha visto grandes cambios económicos, políticos y sociales con la nueva industria. Ésta fue notablemente marcada entre mediados del siglo XVIII y principios del siglo XIX con la que fue la primera revolución industrial, la cual se produjo principalmente en Gran Bretaña y se centró en sustituir las viejas herramientas artesanales utilizadas en la agricultura por máquinas más productivas movidas por energías como el vapor de agua. A partir de ello, el incremento del capital gracias a una mayor producción atrajo grandes fortunas para la inversión, consolidando así una mentalidad burguesa con el objetivo de generar aún más beneficios. También, el comercio del territorio se vio favorecido con la aparición de nuevos medios de transporte terrestre y marítimo como fueron la locomotora o el barco de vapor. Sería ya a finales del siglo XIX cuando, con el comienzo de la segunda revolución industrial, surgirían procesos en cadena basados en nuevas fuentes de energía como el gas o la electricidad para la producción en masa, algo que supondría un cambio total en el paradigma de las relaciones económicas internacionales y las comunicaciones dada la aparición de nuevos medios construidos en acero como el ferrocarril o el automóvil, elevando así la facilidad en el consumo entre la población y la producción de material bélico que se emplearía durante las dos guerras mundiales. Estas transformaciones en la cadena afectaron a los trabajadores en cuanto a la organización y gestión. Sería ya a comienzos del siglo XXI cuando la industria volviese a ver importantes cambios en la automatización de la producción dado el uso de nuevas tecnologías electrónicamente capaces de almacenar energía para procesar la información tales como las computadoras o dispositivos de uso específico. También, con la mayor utilización de energías renovables como la solar o la eólica y nuevas centrales nucleares se posibilitó un crecimiento en cuanto a la demanda tanto en la industria como en otros sectores.

Sería ya a partir de 2016 cuando se iniciase la cuarta etapa de revolución industrial o Industria 4.0[8] con la tendencia hacia una automatización de la cadena de producción basada en en el intercambio de información entre sistemas ciberfísicos o *Cyber-Physical System (CPS)*[9] conectados a través de una *Wireless Sensor Network (WSN)* en lo que se conoce como IIoT[10] para la computación en la nube. Una posible definición de esto podría ser la que proporciona la empresa *Real Time Innovations*[11], donde afirma que «se trata de una infraestructura compuesta por multitud de dispositivos conectados entre sí y donde se puede monitorizar, recoger, intercambiar, analizar y actuar instantáneamente de forma inteligente para cambiar el comportamiento o el entorno sin necesidad de intervención

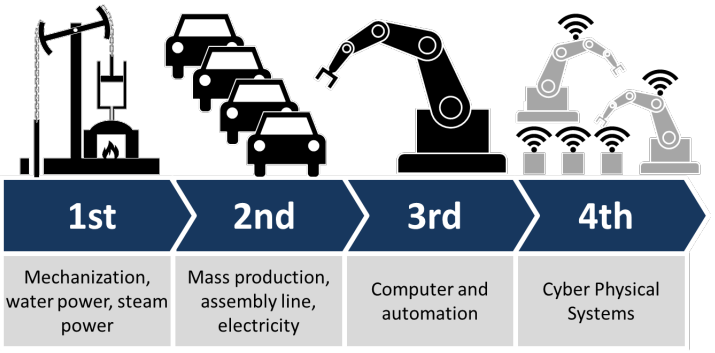


Figura 2.1: Progreso de las diferentes revoluciones industriales

humana». Por tanto, se persigue la customización de los procesos industriales junto con el control automatizado por parte de estos sistemas para unir el mundo físico tal y como lo percibimos con un mundo digital que lo observa desde sus parámetros y que es capaz de tomar decisiones en base a ello, de ahí el nombre de estos sistemas. Otros factores muy importantes en esto son la flexibilidad y escalabilidad de estos sistemas ya que existe una clara preferencia por lograr la movilidad de un gran volumen de dispositivos conectados con otros dispositivos o incluso con personas a través de redes móviles como puede ser el 5G. Un claro ejemplo de esta nueva industria actual puede ser el uso de la robótica como parte de dicho proceso físico, bien sea en la parte de producción mediante actuadores mecanizados con comunicación con un sistema de computación centralizada para la toma de decisiones o en la de la logística asociada en la que pueden aparecer elementos como vehículos guiados para almacenamiento y/o transporte. A raíz de esta Industria 4.0 han surgido otros términos para proyectos relacionados y de enfoque más limitado como son la Energía 4.0, la Construcción 4.0 o la Logística 4.0, pues los objetivos son los mismos: la automatización y la autogestión. También pueden intervenir en el control los sensores del entorno con factores como la temperatura, la humedad, el estado de los dispositivos, la posición y su velocidad, proximidad a objetos, etc. En la Figura 2.2 puede verse la pirámide de automatización derivada del estándar ANSI/ISA-95 por el que se pretende complementar con dicha transformación digital a través de la conectividad total entre los diferentes niveles que hasta entonces se encontraban más jerarquizados en abstracción en el que participan los propios medios de producción, los sensores que comprueban el estado de éstas y del proceso, los *Programmable Logic Controller (PLC)* y actuadores que los controlan y manipulan, herramientas de monitorización con interfaz a usuarios, planificadores de procesos de producción y ya a un nivel más alto las estrategias de negocio. Con ello, se logra una mayor flexibilidad para la extracción de información al interconectar todo (*any-to-any*) y comunicar de forma proactiva los controladores directamente con los sensores y actuadores en un paradigma entre *Edge Computing*, para el procesamiento local de datos con un bajo retardo; y *Cloud computing*, con el pretexto de un control más generalizado a través de elementos de procesado más lejanos distribuidos en la nube. Esto es lo que se conoce como *Fog Computing*[12], donde aparecen nodos intermedios que lidian con aplicaciones más orientadas a IIoT, más ligado a *Cloud* como venimos mostrando; y aquellas más críticas que requieren de un menor retardo. Con todo esto se consigue una convergencia entre la tecnología operacional (OT) y la tecnología de información (IT) en torno a estos CPS interoperables, haciendo así desaparecer la pirámide de automatización con una integración vertical; así como los propios procesos de producción en una horizontal en un todo dada esa automatización en las decisiones.

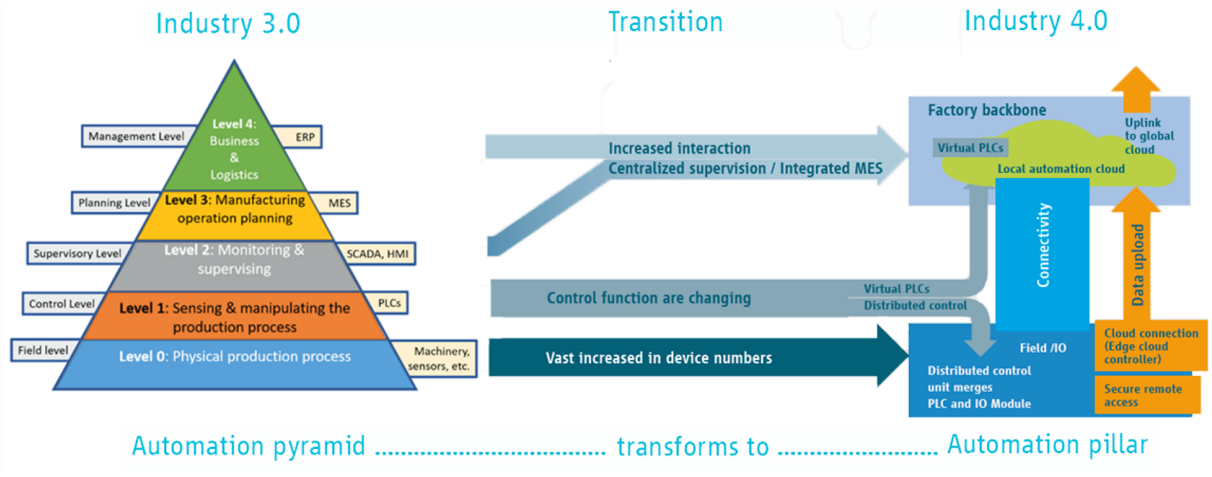


Figura 2.2: Pirámide de Automatización ANSI/ISA-95 [5]

Con esto se puede lograr un modelo de IIoT en el que se satisface la interconexión de toda una cadena de producción a fin de automatizar y optimizar todo el ciclo de procesos, tal y como se puede ver con la Figura 2.3. Nótese que ahora existen elementos que producen datos que se computan en la nube a la vez que también existen otras comunicaciones *Machine-to-Machine (M2M)* para agilizar y reducir el retardo. Es por tanto necesario que toda la infraestructura así como esta información que circula por la red, principalmente aquella destinada a *Cloud* como pueden ser los datos de privacidad de los clientes o las políticas internas de la empresa que aseguran su competitividad en el mercado, esté bien protegida mediante mecanismos de ciberseguridad y resiliencia ante posibles vulnerabilidades. Todo esto supone un gran reto tecnológico y estratégico para las empresas, por lo que esta transformación se debe ir consolidando poco a poco.

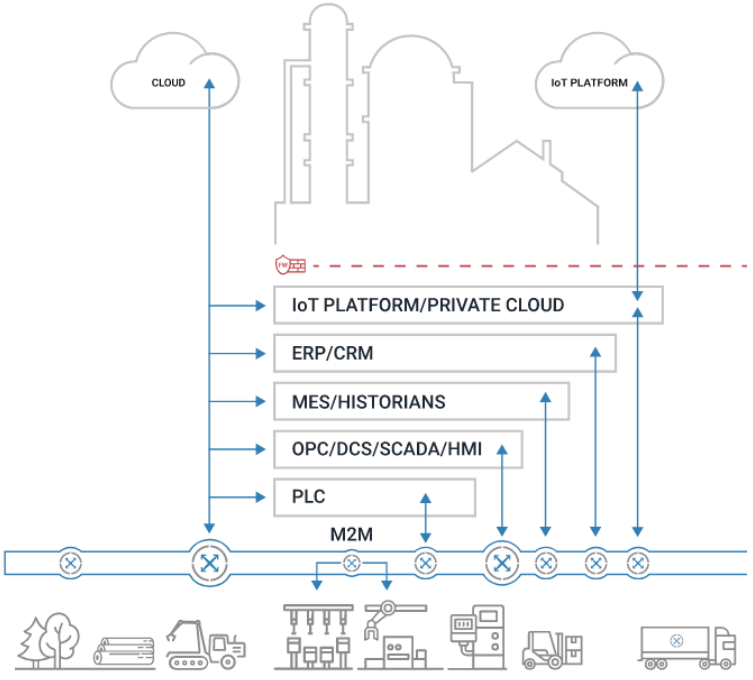


Figura 2.3: Cadena de producción interconectada en Industria 4.0

Su uso abarca un gran número de aplicaciones de acuerdo a los requisitos, desde aquellas que requieren de estricto tiempo real y que por tanto son de mayor criticidad hasta las que poseen restricciones más relajadas según indica el 3GPP en la especificación TS 22.104[13], entre las que se encuentran:

- **Automatización de fábricas**. Se encarga del control automatizado, la supervisión y la optimización de los procesos de trabajo dentro de una fábrica para producciones masivas. Los tipos de servicios que desarrolla suelen presentar los requisitos más estrictos de la infraestructura de comunicación subyacente, sobre todo en términos de disponibilidad, fiabilidad y latencia. Esto incluye controladores de bucle cerrado y robótica que actúan sobre uno o múltiples procesos. Entre estas aplicaciones se encuentran el control de movimiento de las máquinas fijas (e.g. brazo robotizado) y en el que se envían mensajes de forma cíclica y por tanto determinista; comunicaciones entre los distintos controladores según la configuración de la red; control coordinado y guiado de elementos móviles según su posición en el entorno (e.g. robots de transporte móviles autónomos y colaborativos); etc. Además, el hecho de que muchos requieran de bucle cerrado con un controlador implica requisitos más estrictos en cuanto a disponibilidad y retardo determinista para la toma de decisiones. Por ello, su consumo energético está limitado, con lo que la planificación de estas comunicaciones debe ser mediada de forma cíclica y determinista. Es por este motivo por el que se hace uso de redes basadas en Ethernet. Por otro lado existen algunas aplicaciones que proporcionan comunicaciones entre controladores, donde ya puede no requerirse de una baja latencia pero sí de cierta fiabilidad, principalmente si están jerarquizados.

- **Automatización de procesos**. Se encarga del control de la producción y la manipulación de productos tales como químicos, alimentos, líquidos, explosivos, etc. Los tipos de aplicaciones que desarrolla deben presentar un consumo de energía mínimo y gran seguridad para mejorar la eficiencia de los procesos de producción. Sus sensores pueden procesar parámetros como la presión, temperaturas, pH, activos como el estados de válvulas o calentadores, niveles como la presión, inventario, etc. Pese a que suelen ser fijos, dado el gran volumen de sensores en función de las dimensiones de la cadena de producción se hace necesaria además una mayor flexibilidad. Además, dado que de igual forma se requiere de una comunicación en lazo cerrado y con el menor retardo posible, este determinismo se hace crucial.

- ***Human-Machine Interface (HMI)***. Se encarga de permitir la interacción entre usuarios y los distintos elementos encarados de la producción de forma que se pueda tener cierto grado de control y gestión sobre la automatización, abarcando desde los sensores más esenciales a las máquinas de cómputo y servidores. También se pretende la supervisión de dichas máquinas a través de interfaces interactivas, por lo que se necesita también una mayor fiabilidad y un bajo retardo en el enlace. Un ejemplo muy relevante de ello sería el botón de parada de emergencia de la cadena desde un panel de seguridad o aquél botón que obliga al operador pulsar con ambas manos para evitar accidentes laborales. Con ello, vuelve a destacarse la importancia de una comunicación fiable y de baja latencia. Entre estas aplicaciones están los paneles de control móviles en dispositivos de uso común (e.g. *smartphones*, *tablets*, ordenadores, etc) o incluso la realidad virtual y/o aumentada. En los casos en los que la interfaz proporcione también vídeo en tiempo real sobre los elementos será producirá un mayor consumo de ancho de banda.

- **Monitorización y mantenimiento**. Se encarga de la organización interna de la producción sin llegar a generar impacto en los procesos de la cadena, sino que trata de verificar el correcto funcionamiento de los diferentes elementos y con esto optimizar el sistema y prevenir posibles fallos a través de la recolección y análisis de los datos. Normalmente, estas aplicaciones son críticas en cuanto a fiabilidad, pero no en cuanto a latencia.

En definitiva, existen multitud de aplicaciones en la Industria 4.0 que requieren de una comunicación fiable y de muy baja latencia en una red que permita la movilidad de los diferentes elementos que intervienen en la cadena de producción así como una gran flexibilidad en cuanto al número de dispositivos que se conectan a ésta. Es por ello por lo que, como puede verse en la Figura 2.4, existe una creciente evolución del uso de la tecnología industrial de Ethernet, pero también de las redes inalámbricas.

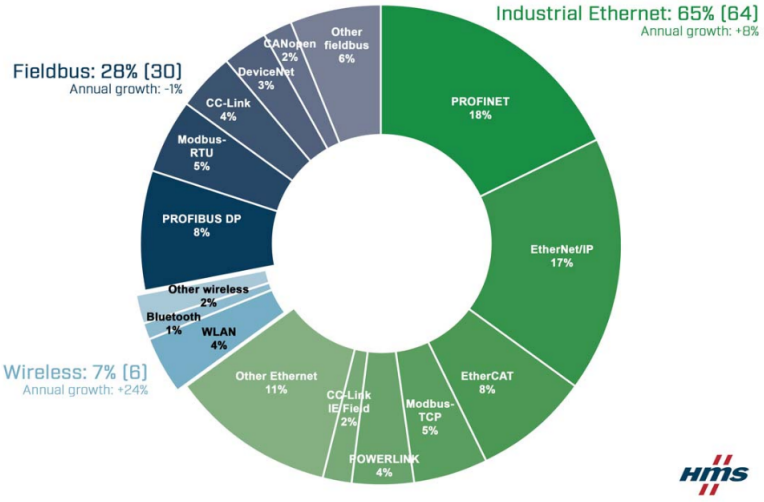


Figura 2.4: Crecimiento de las tecnologías inalámbrica y Ethernet en la Industria 4.0

Con esto, la nueva generación estandarizada de redes móviles 5G parece ser una gran candidata para su uso en el sector. De hecho, esto no es casualidad dado que ya desde la anterior generación se buscaba la forma de poder interconectar en masa los dispositivos de forma que pudieran interactuar entre ellos de forma casi instantánea, como es el caso de las aplicaciones de tiempo real ya comentadas. Es precisamente por este motivo por el que muchos de los *Mobile Network Operator (MNO)* han decidido invertir en infraestructura de proximidad que proporcione a las empresas de esta industria tal conectividad a través de segmentos privados destinados únicamente a tales aplicaciones mediante *network slices*, lo cual lo lleva a convertirse también en un negocio de transformación de las telecomunicaciones para la mejora de la calidad de servicio o *Quality of Service (QoS)* y en el que aparecen ideas basadas en *Cloud* como la Inteligencia Artificial, el *Big Data*, etc. Con ello, el estándar no ha pretendido únicamente aumentar el ancho de banda destinado a comunicaciones móviles o eMBB, como venía ocurriendo hasta entonces con el objetivo de llegar a un usuario final, sino que también se ha centrado en perseguir estos otros en el sector de la industria tales como la mejora en productividad, una mayor flexibilidad del sistema, asegurar la calidad del producto, la anticipación en las decisiones, reducir su *time-to-market*, involucrar a la empresa en el proceso, nuevos modelos de negocio, prevención de errores, etc.; con comunicaciones en las que ya apenas interactúan las personas y

con las que se producen latencias radio por debajo de 1 ms y con una alta fiabilidad de las transmisiones, de hasta el 99,9999 % en algunos casos. Un claro ejemplo de ello pueden ser también la conducción autónoma de vehículos o el *platooning*, donde los vehículos inteligentes son capaces de interactuar entre ellos para optimizar el gasto de combustible, la seguridad en las intersecciones, etc. Éstas son las comunicaciones URLLC y en las cuales nos centramos en este proyecto. Por ello, este determinismo de las redes industriales parece ser fundamental, por lo que el uso de tecnologías estandarizadas en el segmento cableado como es el caso de TSN en la capa de enlace se ha convertido en la solución más extendida para comunicaciones en tiempo real más orientadas a esta Industria 4.0. Dado que estas comunicaciones se realizan generalmente de manera cíclica, la opción más razonable es la de TSN síncrono al ser aquélla en la que existe sincronización entre el conjunto de la red y los dispositivos de modo que se puedan transmitir una serie de paquetes en un determinado instante y que éstos atraviesen la red con el menor retardo posible. A continuación se pasa a estudiar tanto TSN como el sistema de 5G y la integración entre ambos sistemas.

2.2. Time-Sensitive Networking

TSN[5] es una extensión definida por la asociación de estándares *Institute of Electrical and Electronic Engineers (IEEE)*, en concreto por el grupo de trabajo 802.1Q el cual engloba aquellos destinados al encapsulado Ethernet (capa de enlace) en medios compartidos de redes VLAN. Se creó con la intención de proporcionar determinismo y fiabilidad para las aplicaciones sobre redes OPC-UA basadas en Ethernet donde se requiere de comunicaciones de tiempo real tales como las ya mencionadas en la Industria 4.0, aunque también sobre la red propietaria ya desplegada, por lo que existen múltiples alternativas que hacen uso de TSN como PROFINET, EtherCAT o SERCOS. Por tanto, a diferencia del resto del consumo de redes Ethernet donde se mira más por el ancho de banda, en TSN se tiene como objetivo minimizar el retardo y el *jitter* de paquetes que se transmiten en instantes de tiempo concretos. Esta tecnología permite clasificar en cada puerto de salida los diferentes flujos de tráfico según los requisitos de cada uno para cumplir cierta QoS. De este modo, un controlador decide qué cola se transmite en cada momento en función de estos parámetros a fin de establecer prioridades a través de un identificador, éste es el *Gate Control List (GCL)*. Este esquema puede verse en la Figura 2.5.

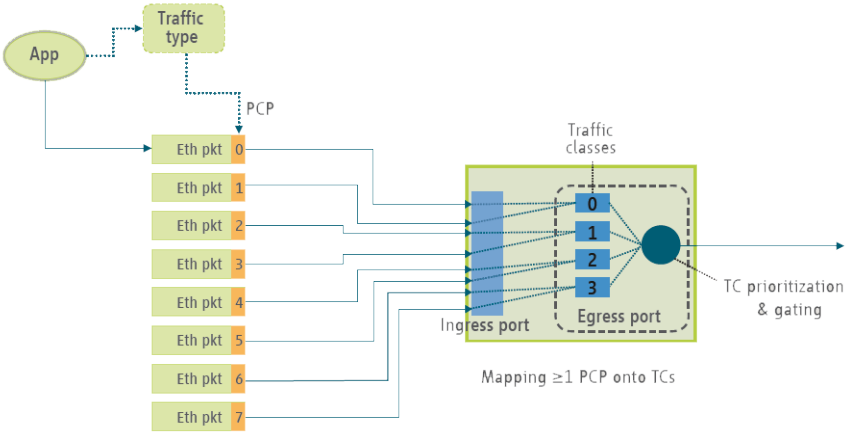


Figura 2.5: Sistema de prioridades y clases en TSN [5]

Por tanto, según las políticas de QoS de la red TSN, asignará a cada flujo un identificador *Priority Code Point (PCP)* en función de su criticidad, como podría ser la señalización de control periódica o la autogestión de la propia red. Normalmente se contemplan un total de ocho, con valores desde el 0 hasta el 7. Estos identificadores pueden clasificarse de forma individual o en grupos según el número de colas o clases configuradas en el puerto de salida, es decir, una vez los paquetes ya han sido conmutados. Con ello, si cada valor de PCP se relaciona con una clase distinta, podrá ser tratado de forma única acorde a dichas políticas para este flujo, aunque puede interesar que en cierto segmento de la red se agrupen para garantizar el correcto funcionamiento del tráfico. Para poder llevarse todo esto a cabo, la red TSN suele estar gestionada de forma centralizada por un controlador, tal y como se muestra en la Figura 2.6.

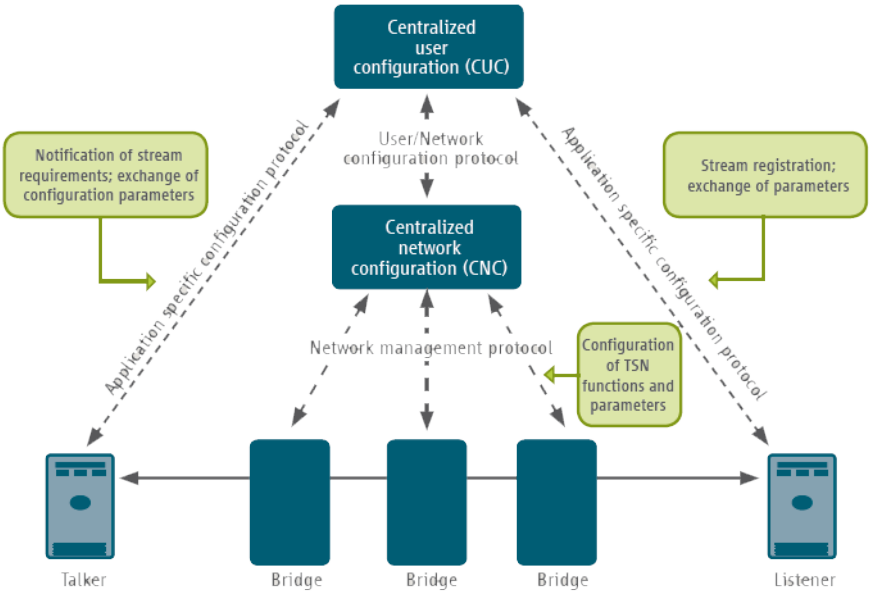


Figura 2.6: Sistema de control en TSN [5]

De este modo, TSN actúa de forma centralizada a través de lo que se conoce como *CNC* en un paradigma similar a SDN. El CNC es el encargado de establecer las configuraciones de cada uno de los nodos o *bridges* que forman la topología completa de la red TSN y recibir los reportes que éstos le proporcionan, tales como el número de prioridades y clases soportadas, los retardos en función del puerto y de la clase con el siguiente salto, las políticas de los flujos ya configurados, etc. Por otro lado, *Centralized User Configuration (CUC)* tiene comunicación con los dispositivos finales, tanto el transmisor o *talker* como el receptor o *listener*; y cuyas tareas están más orientadas a procesar los requerimientos QoS de la comunicación entre ambos extremos, comunicárselos al CNC a través de la interfaz red-usuario para que éste determine las configuraciones necesarias en base a ello como pueden ser los tiempos de inicio y final de la transmisión, el PCP para cada flujo, el ancho de banda necesario, el retardo promedio estimado, etc; y finalmente transmitírselas a estos dispositivos extremos a fin de establecer la comunicación en la ventana de tiempos adecuada. Es por este motivo por el que tanto CUC como CNC permanecen continuamente interconectados para que este último pueda aplicar las configuraciones sobre toda la topología involucrada acorde a las necesidades de cada uno y en función de los parámetros que caracterizan a los nodos y que éstos reportan al CNC. Existen sistemas en los que estas capacidades de configuración se distribuyen entre los dispositivos finales y los nodos

de la red, diluyéndose así estas entidades CNC y CUC, aunque suelen ser menos comunes en las redes industriales dado que se necesita que la topología sea dinámica ante cambios, sobre todo si se pretenden integrar con redes inalámbricas con canales cambiantes como es el caso de 5G; además de que se saturaría más la red a través de la señalización de control entre los nodos.

Dado que Ethernet se trata de una tecnología de acceso compartido al medio por *Time-Division Multiple Access (TDMA)*, estas prioridades son las que definen el orden y los tiempos en la asignación de los recursos para las distintos flujos que discurren a través de los enlaces. Existen diferentes casos en función de cómo se realiza esta reserva a fin de mantener una red determinista, a continuación se definen brevemente los más relevantes:

- **Síncrono** (IEEE 802.1Qbv). Se planifican los tiempos asignados a cada clase, puerto y nodo a través de una misma referencia temporal. Esto se logra a través de *gPTP*, midiendo la diferencia entre un reloj maestro y el marcado por el nodo con precisiones por debajo del microsegundo para así poder definir cada intervalo de tiempo para cada cola a través de lo que se conoce como *Time-Aware Shaper (TAS)* y con ello el retardo de los paquetes para reportar su estado al CNC. Por tanto, dichos tiempos quedan definidos a partir de un instante inicial y otro final de forma cíclica sobre un «híperperiodo». Con ello, las puertas de cada cola se abrirán y cerrarán en función de estos tiempos predefinidos en la lista GCL. Esta lista será esencial para tratar de optimizar los tiempos asignados a cada cola ya que se irán adaptando en función de la situación (e.g. tráfico para cada PCP, número de clases, periodicidad de éstas, retardos, etc.). Tras esto, los siguientes nodos ajustarán también acordemente sus tiempos en su propio híperperiodo de forma que el retardo sea el menor posible en esquemas de agregación cíclica. Sin embargo, todo esto implica reducir la capacidad de los enlaces para la transmisión de estas señales de sincronización.

- **Cíclico** (IEEE 802.1Qch). Se trata de una versión más simple del caso síncrono, en el que tan solo se realiza una secuencia repetida de selección una cola u otra, independientemente del estado de estas colas ya que no se ha realizado previa planificación. Normalmente, se prefiere el uso de tal planificación con el fin de optimizar el tiempo de las tramas en las diferentes colas.

- **Asíncrono** (IEEE 802.1Qbu). Las transmisiones de las diferentes colas ya no se planifican por tiempos de forma sincronizada entre nodos, sino que se implementa un algoritmo destinado a seleccionar cada una a través de lo que se conoce como *Urgency-Based Scheduler (UBS)*. Éste filtra y clasifica los flujos en diferentes fases en función de la tasa y de la prioridad con la que llegan las tramas, es decir, se aplican filtros a *buckets* a cada uno en función de la situación que éstos presenten para posteriormente asignarles una prioridad interna IPV que los clasifica en una cola de estricta prioridad o en otras que pueden ir alternándose. De este modo, tan pronto como llegue un paquete de estricta prioridad, éste es transmitido a la vez que se interrumpe la transmisión de la trama de menor prioridad (IEEE 802.1Qbr), lo que se denomina anticipo o *preemption*. Mientras tanto, el resto de colas (aunque suele ser única) son las que se transmiten. Esto se consigue mediante el uso de preámbulos al principio y al final del anticipo para delimitar la acción de modo que el receptor pueda detectarla. Para que se pueda producir el anticipo, es necesario que la trama de menor prioridad que ya ha comenzado a transmitirse sea menor a 64 Bytes, de lo contrario la de mayor prioridad aguardará en cola. El problema de este sistema es que los filtros de por sí ya introducen un retardo considerable y puede afectar

notablemente a los requerimientos del resto de flujos, ya que además está insertando una sobrecarga con tales preámbulos. Todo este esquema de prioridades por anticipo se denomina *Asyncrhonous Traffic Shaper (ATS)*.

- **Basado en crédito** (IEEE 802.1Qav). Se emplea un valor numérico como crédito de cada cola, el cual aumenta a medida que van llegando las tramas a ella y quedan esperando en la cola y en proporción a su tamaño. Aquella cola con mayor crédito en un determinado instante es la que se transmite, pero a cambio de ver reducidos sus créditos en función de la velocidad del puerto. El gran problema de esto es que no se transmite nunca mientras su valor esté por debajo de cero, con lo que aumenta los tiempos de espera y pueden provocar el incumplimiento de los requisitos de tales flujos si estos no son más relajados.

Por tanto, la gran ventaja del esquema asíncrono es que no es necesario el uso de un reloj compartido entre los nodos como en el caso del síncrono, lo que lo hace más escalable y con una mayor capacidad en los enlaces, aunque la complejidad para seleccionar cuáles se deben transmitir de forma inmediata aumenta a medida que aparecen nuevos flujos o la topología evoluciona. Sin embargo, muchos dispositivos de la Industria 4.0 transmiten y reciben la información con una periodicidad muy estricta necesitando así una misma referencia de tiempo precisa para disminuir el retardo, por lo que en redes industriales parece más razonable elegir la opción síncrona. El esquema asíncrono podría ser una gran opción en el caso de un tráfico más esporádico y sencillo.

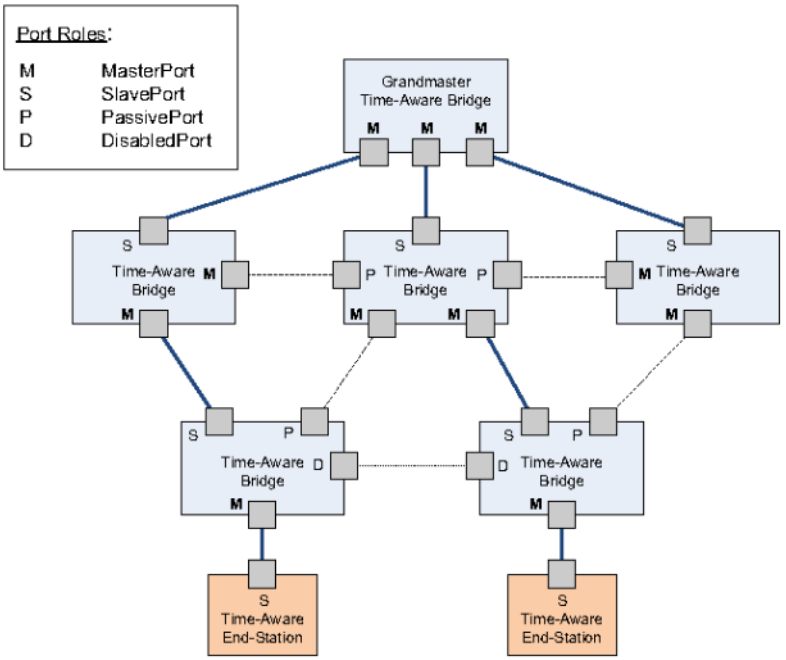


Figura 2.7: Jerarquía de sincronización de relojes en IEEE 802.1AS

Para la sincronización[14] se hace uso del estándar IEEE 802.1AS, el cual define el protocolo gPTP previamente mencionado, generalizando así el protocolo anterior IEEE 1588 para redes Ethernet deterministas con prioridad. Con ello, se mide el tiempo de residencia de un paquete en cada conmutador TSN, es decir, desde que un paquete llega hasta que se transmite junto con la latencia del propio enlace hasta el siguiente salto. Estos tiempos se calculan por encima de la capa *Medium Access Control (MAC)* a través

de la referencia con el reloj maestro o *Grand-Master (GM)* hasta los diferentes nodos y dispositivos finales que participan en la red. La exactitud de este sistema dependerá principalmente de la precisión de las medidas del tiempo de residencia y enlace que se realicen en cada nodo, haciendo así uso de una relación entre la frecuencia entre el reloj de dicho nodo y el del GM para poder medir correctamente dicho retardo a fin de garantizar los requisitos. Para ello, el protocolo decide cuál será el GM de entre todos los nodos así como la jerarquía entre los puertos de los diferentes nodos a través del algoritmo *Best Master Clock Algorithm (BMCA)*. De esta forma, el GM actuará como la fuente de referencia temporal y los puertos se dispondrán de forma jerárquica *master-slave*, donde aquél puerto más cercano al GM será el *Clock Master (CM)*, es decir, el que propaga la señal de reloj; mientras que el más cercano al dispositivo a sincronizar será el *Clock Slave (CS)* siendo así el que la recibe, tal y como se muestra en la Figura 2.7. El resto de puertos permanecerán pasivos o deshabilitados en este *spanning tree*, según se configure. Puede configurarse más de un dominio de reloj de modo que estos puertos actúen con un rol u otro bajo una misma referencia de tiempo.

Para que esto ocurra, el CM envía al CS un mensaje de sincronización *Sync* en el que figura el tiempo de referencia de salida del GM y el campo de corrección del retardo que éste ha calculado sobre sí mismo y el acumulado de los saltos previos. Además, con él marca el instante de tiempo de salida *t1* y con el que el CS obtiene un instante de llegada *t2* para luego recibir también un segundo mensaje de *Follow-Up* ya con el valor de *t1*, siendo esta diferencia de tiempo la suma del *offset* entre relojes más la de propagación por el enlace. Tras ello, el CS enviará al CM el mensaje *Delay Request* con el fin de marcar el tiempo *t3* y que este último haga lo mismo cuando lo reciba en el instante *t4* para luego así comunicárselo al CS con un mensaje de *Delay Response*. De este modo, el reloj del CS podrá calcular con estos cuatro instantes el retardo que introduce el enlace como la mitad de la suma de ambas diferencias (*t2-t1* y *t4-t3*). Con ello, ya podrá conocer ese *offset* a partir de éste y ajustará la relación con la frecuencia del GM añadiéndolo al campo de corrección. Este proceso puede verse con la Figura 2.8.

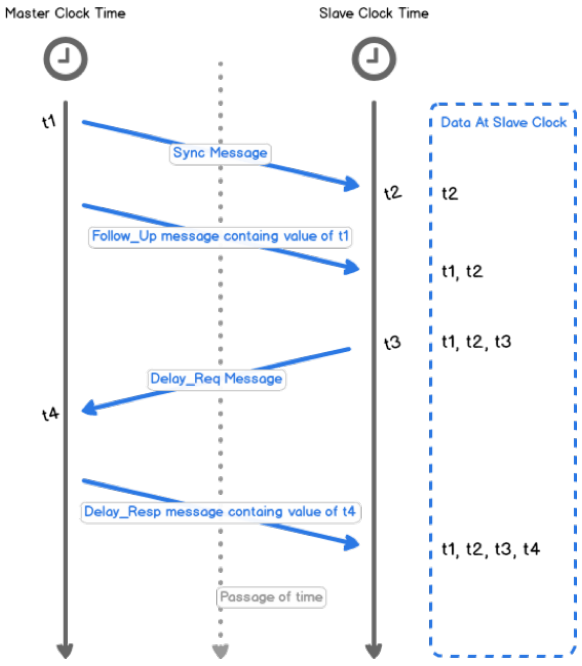


Figura 2.8: Sincronización mediante mensajes de tiempo en gPTP [14]

Por otro lado, para conocer el tiempo de residencia, basta con calcular la diferencia entre las marcas de tiempo de un paquete de llegada a un puerto CS hasta que lo retransmite por su puerto CM para añadirlo también al campo de corrección y transmitirlo al siguiente nodo. Nótese que, como se ha comentado antes, este procedimiento de sincronización a través de mensajes se realiza de forma periódica, por lo que consumirá parte de los recursos de la red como es la capacidad de los enlaces. Esta referencia de tiempo del GM puede ser a través de un reloj local o mediante uno universal externo, aunque en este último caso se debe tener en cuenta su procedencia así como las posibles variaciones entre señales si pertenece a un dominio fuera de la red determinista, por lo que esta solución simplifica el esquema siempre y cuando el requisito de precisión en la sincronización sea más relajado. También existen nuevas versiones este protocolo con enmiendas para soportar modelos como YANG, *half-duplex*, etc.

Con ello, existen otros estándares diferentes de IEEE que también forman TSN síncrono. Éstos pueden dividirse según su naturaleza [15]:

- **IEEE 802.1AB**. *Link Layer Discovery Protocol (LLDP)*[16] se utiliza para comunicar las capacidades y el estado actual del nodo al CNC de modo que éste conozca bien la topología con la que va a operar, así gestionando sus recursos a través de *Simple Network Management Protocol (SNMP)* para lograr el mínimo retardo extremo a extremo. Para ello, varios agentes LLDP se distribuyen entre los *switches* TSN con el objetivo de comunicarse con un manager LLDP ubicado en el CNC, el cuál se encarga de recopilar toda esta información que los agentes le proporciona tal como la VLAN a la que pertenecen, identificador del puerto, información MAC, agregación, capacidades de conmutación, etc. Esto se realiza de forma periódica para cada uno de los puertos. Además, cada agente cuenta con una base de datos local con la que almacena la información sobre su propio estado y capacidades en instancias TLV (*Type*, *Length*, *Value*); así como una remota en la que recibe y almacena información proporcionada por sus nodos vecinos.
- **IEEE 802.1AX**. Permite definir lo que se conoce como *Link Aggregation Group (LAG)*, es decir, un subconjunto de flujos paralelos que hacen uso de los mismos recursos de la red de modo que puedan ser tratados de forma lógica en un mismo enlace entre nodos seleccionando dinámicamente así la configuración que más convenga para adaptarse a las condiciones de la red. Para ello es necesario que la suma de anchos de banda de todos ellos sea siempre inferior al límite de capacidad del enlace físico compartido, lo cual se puede suplir añadiendo más puertos destinados a dicho enlace. La resiliencia de estas redes son muy importantes dado que si un nodo cae, otro debe suplirlo, por lo que normalmente se hace uso de sistemas como *Rapid Spanning Tree Protocol (RSTP)*. De este modo, dichos flujos se pueden también duplicar o balancear entre nodos para ganar en fiabilidad.
- **IEEE 802.1CB**. También conocido como *Frame Replication & Elimination Reliability (FRER)*, permite el envío de copias de cada trama TSN a través de rutas diferentes con el fin de proporcionar redundancia proactiva para aplicaciones de alta fiabilidad en las que no se pueda tolerar pérdidas, como podría ser el caso de que un nodo cayera y automáticamente ésta se descartara. Para evitar mensajes duplicados, cada trama posee un identificador de forma que si el destino ya lo ha recibido se elimine la última copia en la red. El principal problema que acarrea esta funcionalidad es la mayor congestión de la red en el caso de que exista un gran número de flujos entre estas aplicaciones.

- **IEEE 802.1CS**. *Link-local Registration Protocol (LRP)* permite la replicación de cambios de una base de datos de un nodo a la de otro conectados a través de un enlace para mantener los registros actualizados para así poder distribuir la información entre toda la red.
- **IEEE 802.1Qca**. El protocolo *Path Control and Reservation (PCR)* se utiliza como una extensión de *Intermediate System to Intermediate System (IS-IS)* y *Shortest Path Bridging (SPB)* con el fin de establecer las posibles rutas explícitas de forma óptima y redundada a un flujo (VLAN) así como de asignarle un ancho de banda determinado en función de restricciones, supervisar su sincronización con los otros nodos y emplear FRER en el caso de haberlo configurado. Este camino explícito es seleccionado por un *Path Computation Element (PCE)*. En el caso de fallo, posee una rápida convergencia.
- **IEEE 802.1Qcc**. Este estándar[17] introduce mejoras respecto al ya existente *Stream Reservation Protocol (SRP)* con una interfaz usuario-red o *User-Network Interface (UNI)* proporcionada por el CNC y que se comunica con los conmutadores con el fin de aplicar nodo a nodo la configuración necesaria y realizar con ello la reserva de recursos (e.g. ancho de banda, planificación temporal, colas, etc.). Por tanto, a través de este protocolo se pueden configurar los TAS de todos los nodos que correspondan a la red TSN con todo el conocimiento posible de ésta al estar el CNC centralizado, como ya se comentó anteriormente.
- **IEEE 802.1Qci**. Con *Per-Stream Filtering and Policing (PSFP)* se consigue realizar un filtrado del tráfico de forma individualizada para cada flujo con el objetivo evitar el mal funcionamiento por sobrecarga en la red debido a las condiciones (número de flujos, valores y proporciones de tráfico por cada PCP, anchos de banda, etc.) o por posibles ataques. En él se aplican reglas SDN de forma que las coincidencias permitan crear *buckets* de flujos con niveles de prioridad específicos y, de lo contrario, aplicar acciones basadas en las políticas establecidas, como podría ser el descarte de una réplica por poseer un retardo superior al requerido o el cambio a un PCP de mayor prioridad si todavía conserva algo de margen hasta el límite. De este modo, el CNC puede definir estas reglas para ajustar además las prioridades con las que un flujo llega en función de las estadísticas de retardo en el destino. Estos filtros no pueden ser muy complejos ya que introducen de por sí mayor retardo en su procesado.
- **IEEE 802.1Qbv**. Este estándar es el que define el TAS anteriormente visto. Este planificador cíclico es sensible al tiempo al contar con la sincronización proporcionada por el protocolo gPTP a partir de un reloj compartido, con lo que para cada intervalo dentro de lo que denominamos ciclo o híperperiodo el puerto elige entre una de las hasta 8 colas a través de su PCP de forma preestablecida según los requerimientos de las aplicaciones que hace uso de la red TSN recogidos por el CUC. Con ello, los tiempos de inicio y final de la transmisión de cada una de ellas, y que se han procesado previamente en el CNC, se ajustan a los de transmisión de los dispositivos de modo que el retardo de las tramas que atraviesan la red TSN sea mínimo. Para lograr esto, el TAS calcula dicho híperperiodo como el Mínimo Común Múltiplo (MCM) de todos los periodos con los que se recibe un PCP clasificado y lo divide en múltiples fases con el Máximo Común Divisor (MCD).

Por tanto, las tramas de una cola se transmitirán periódicamente durante un tiempo resultado de realizar una planificación nodo a nodo desde el origen hasta el destino en

función de su tamaño, el camino seguido por éste y las velocidades de trasmisión de los nodos TSN que lo forman. Este factor así como qué segmento inicial serán los que el CNC finalmente designe en función de las necesidades de la red con un intervalo temporal para cada cola. Ambos valores serán fijos para todas las fases mientras los parámetros del estado de la red sean los mismos. Con ello, el clasificador del conmutador para cada puerto seleccionará la cola correspondiente para cada flujo, así dividiendo el tráfico de llegada en diferentes prioridades organizadas mediante un PCP, es decir, en distintas VLAN. Estos conceptos pueden verse con la Figura 2.9.

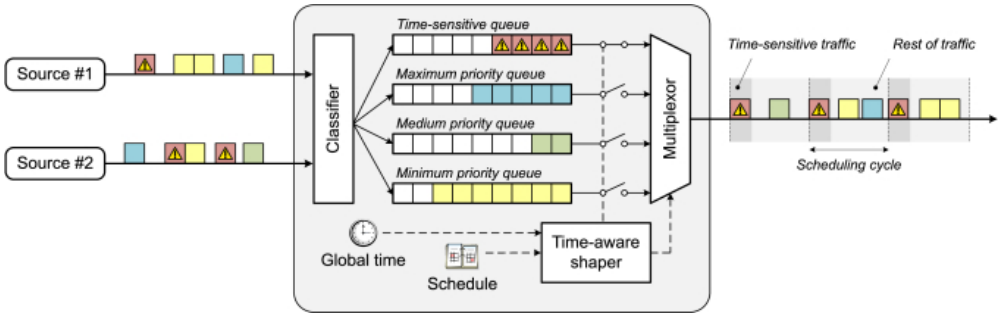


Figura 2.9: Esquema de funcionamiento del estándar IEEE 802.1Qbv

De este modo, el TAS, con la señal de reloj sincronizado con el resto de la red TSN así como la lista de control de puertas (GCL) y sus respectivos estados, irá seleccionando para cada instante una única puerta que conecte a la cola durante el intervalo que correspondan a partir de dicho instante con el multiplexor hacia puerto de salida. Con esto, el estado de la puerta lo define con un valor binario, es decir, entre 1 (abierto) y 0 (cerrado). Todo ello se realiza en la subcapa software *Link Layer Control (LLC)*, por lo que este multiplexor será el que lo entregue a la subcapa hardware MAC para que la información sea procesada y posteriormente codificada. Como resultado, se tendrá la transmisión TDMA de todas ellas tal y como el CNC ha establecido para cumplir con las restricciones que favorecen el cumplimiento del retardo y *jitter* dentro de la QoS para cada flujo. En cambio, mientras no se esté transmitiendo un PCP, las tramas pertenecientes a éste permanecerán en una sección de memoria designada a su cola durante hasta que finalmente sean transmitidas, acumulando así un retardo hasta llegar al destino. Es por ello por lo que el CNC debe jugar con la asignación de estos tiempos así como con el PCP asignado a cada flujo. Un esquema que resumen el funcionamiento de TAS para varias clases en varios nodos es el que se presenta a continuación con la Figura 2.10. Sin embargo, para evitar colisiones debido a ligeros desajustes en el tiempo de transmisión entre la última clase previa del híperperiodo y la sensible al retardo que le precede, se introduce un intervalo de guarda con el fin de permitir que la última trama de tráfico no prioritario se termine de transmitir y que por tanto no afecte al retardo de la otra. Esto requiere por supuesto que dicho intervalo sea suficiente pero no excesivo e inmediatamente previo a la transmisión de la cola de máxima prioridad. Normalmente suele ser de 12 μs, resultado de dividir el *Maximum Transmission Unit (MTU)* de Ethernet (1500 Bytes + 42 Bytes) entre 1 Gbps. Por tanto, los huecos que no se utilicen para ello podrían ser usados para el tráfico *best-effort* de la cola menos prioritaria siempre y cuando en éstos quepa al menos, en el peor de los casos, la duración de la transmisión de la trama de mayor tamaño en Ethernet más una última a modo de banda de guarda, de forma que si una de tal tamaño ya ha comenzado a transmitirse en el último instante antes de cerrarse la puerta de la cola correspondiente no afecte a los tiempos reservados de los flujos de mayor prioridad. Durante estas bandas de guarda no

se transmite ninguna cola, lo que implica en cierto modo desaprovechar los recursos para garantizar su estabilidad. Entre colas no sensibles al retardo no es necesario añadir estos intervalos de guarda, salvo que existan fluctuaciones como se verán en el Apartado 2.4. Es muy común por tanto hacer uso de dos únicas colas: la de tráfico sensible al tiempo y la del no planificado, aunque puede ampliarse este número de colas según se pretenda dar mayor prioridad a un tipo de tráfico u otro tal mediante el tiempo o la frecuencia de apertura. Este trabajo se centra de momento únicamente en el tráfico que requiere cumplir con unos requisitos temporales, tanto en cuanto al retardo máximo con el que pueden llegar a su destino como a lo que su periodo de transmisión se refiere; y en optimizar el aprovechamiento de los recursos.

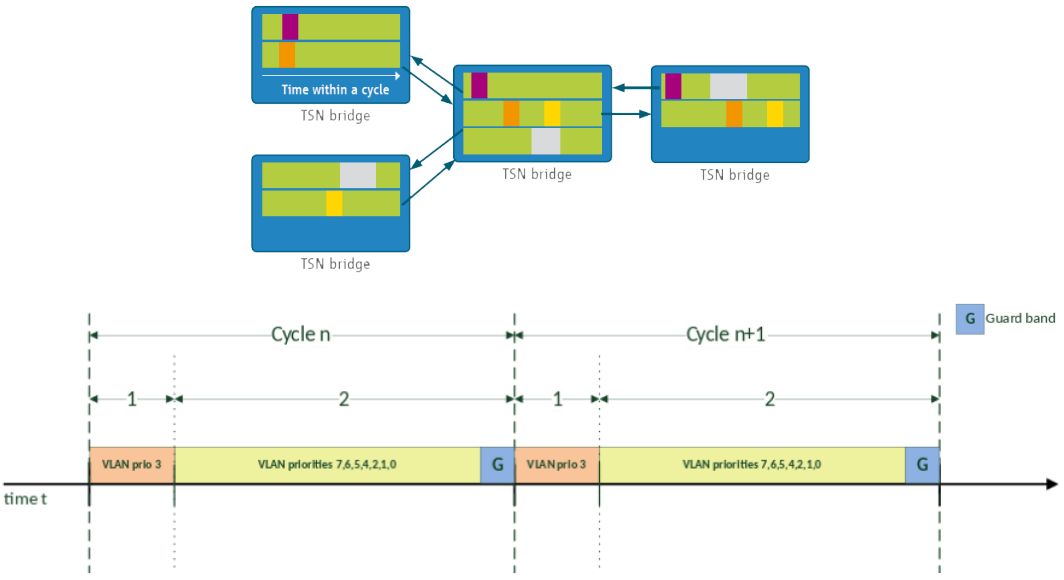


Figura 2.10: Funcionamiento de TSN síncrono (a) entre nodos [5] (b) ciclos del nodo

Normalmente se establece un PCP en función del tipo de tráfico y su criticidad (e.g. eventos de control, gestión, vídeo, señales de estado, *best-effort*, etc.). Sin embargo, en la Industria 4.0, se puede aprovechar todo esto de modo que diferentes flujos sensibles al retardo se puedan agrupar en diferentes valores de PCP para cumplir con un retardo u otro según su valor de retardo máximo. En el caso opuesto, es posible que requieran reagruparse en una misma clase mediante PSFP con el fin de dar cabida a otro tipo de tráfico en un determinado segmento de la red, aunque este esquema es bastante más complejo de tratar. Se debe tener en cuenta de que, en el caso de que más clases se utilicen para este tipo de tráfico sensible al retardo, se tendrán que añadir nuevos intervalos de guarda que antecedan dichas transmisiones.

Para comprender todo esto un poco mejor, se muestra la Figura 2.11 en la que ya aparece toda una red determinista TSN perteneciente a la Industria 4.0 para dar conectividad a sensores y actuadores de modo que los controladores PLC sean capaces de dar respuesta ante sus estados en el menor tiempo posible dentro de las aplicaciones críticas que venimos comentando. Para ello, se integran una serie de nodos especialmente destinados a la conmutación de mínimo retardo sobre una red Ethernet ya desplegada, lo que se conoce como un escenario *brownfield*. De este modo, se observa una topología centralizada en la que aparecen tanto el CNC como el CUC en el *Fog*, controlando así la red a través de los enlaces y nodos puramente Ethernet en la misma red troncal que aquellos destinados a

TSN. Aparecen además hasta cuatro dominios de reloj diferentes que parten desde uno global, por lo que los diferentes dispositivos se adhieren a uno u otro además del global en función del segmento de red en el que se encuentren. Estos dominios son temporales y no TSN, caso en el que cada uno de ellos tendría que poseer sus propios CNC y CUC y jerarquizados entre ellos.

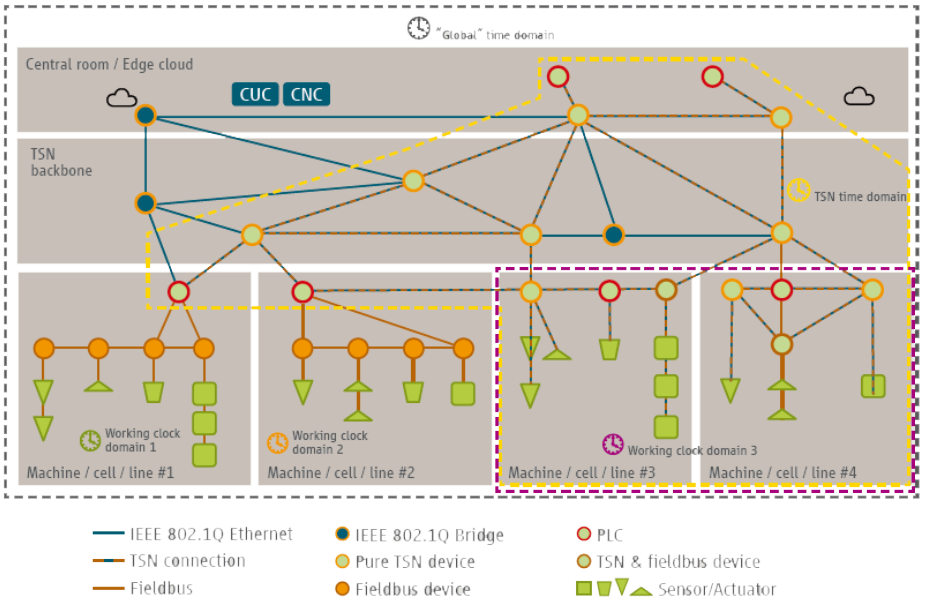


Figura 2.11: Red TSN completa en la Industria 4.0 [5]

Sin embargo, el principal problema que acarrea esto es el gran coste que conlleva cablear todas las conexiones que comunican los dispositivos con el resto de la red y los controladores. Además, muchos de los dispositivos carecen de la movilidad necesaria para ejecutar determinadas acciones al verse limitados por dicho cableado. Este problema crece de forma exponencial cuanto más compleja sea la topología de la red y más comunicaciones M2M se requieran. Es por ello por lo que se propone el uso de una red inalámbrica como es la nueva generación 5G.

2.3. Sistemas 5G NR

El estándar 5G *New Radio*[18] se ha consolidado como la nueva generación de redes móviles destinada a la mejora de las capacidades para albergar un mayor tráfico más heterogéneo. Hasta ahora, las generaciones anteriores se habían centrado en las comunicaciones móviles tradicionales (eMBB) para albergar un mayor número de usuarios con mayores ancho de banda, velocidad, disponibilidad, etc.; pero con 5G se ha perseguido además, desde el primer momento, ser aquélla que también proporcione comunicaciones en el demandado mundo de *Internet of Things (IoT)* más enfocado a la industria para una mayor flexibilidad con interconexiones de multitud de dispositivos de bajo consumo distribuidos (e.g. CPS) y con la nube, con bajas latencias y una alta fiabilidad principalmente en la parte radio y con la debida adecuación de la red troncal. Con esta tecnología, ahora ya se hace posible la implementación de la nombrada Industria 4.0. Adicionalmente, pretende elevar dicha velocidad de usuario desde al menos los 100 Mbps hasta los 10 Gbps y con ello multiplicar por 10.000 su capacidad respecto a la generación anterior. Dichos requisitos fueron publi-

cados en una recomendación por primera vez en septiembre de 2015 por la *International Telecommunication Union (ITU)* a través del estándar *International Mobile Telecommunications (IMT)* M.2083[19], donde ya se habían establecido previamente la tercera y cuarta generación de telefonía móvil; y recogidos por el consorcio del 3GPP con la Release 13 para la continuación del desarrollo del estándar ya desplegado de LTE-Advanced, aunque se trataría de un nuevo sistema *New Radio (NR)* no retrocompatible con las tecnologías anteriores a éste. Poco más tarde surgiría el *5G Infrastructure Public-Private Partnership (5G-PPP)* en Europa con el fin de asegurar un liderazgo en el continente respecto a la relación entre este estándar y la Industria 4.0 de la que venimos hablando hasta ahora enmarcada en el mercado global. Este nuevo estándar describe 3 posibles escenarios de uso:

- ***Enhanced Mobile BroadBand (eMBB)***. Contempla la mejora en el rendimiento y de la experiencia de usuario en relación a las generaciones anteriores tanto para los entornos cerrados y rurales como para nuevos casos de mayor densidad tales como pueden ser los macroeventos urbanos. Ofrece picos de hasta 20 Gbps en el enlace descendente y de 10 Gbps en el ascendente con una mayor eficiencia espectral de hasta 0,3 bps/Hz y movilidad entre las celdas de cobertura de la estación base, así permitiendo el uso de aplicaciones más accesibles y demandantes como vídeo de alta definición, servicios *Cloud*, realidad aumentada, etc.
- ***Massive Machine-Type Communications (mMTC)***. Debe soportar una gran cantidad de dispositivos conectados que tradicionalmente transmiten un menor volumen de datos no sensibles al retardo. Con la aparición de IoT su uso se ha generalizado a dispositivos de bajo coste y menor consumo, por lo que debe ser capaz de albergar un gran volumen de tráfico de entre todos ellos, entre los que se pueden encontrar sensores y actuadores, sistemas de videovigilancia, *wearables*, etc.
- ***Ultra-Reliable Low-Latency Communications (URLLC)***. Existen aplicaciones en las que sí existe una exigente sensibilidad al retardo con una gran fiabilidad, principalmente en procesos de producción en la Industria 4.0, cirugías médicas en remoto, vehículos de conducción autónoma, *Tactile* Internet, etc.; por lo que se debe satisfacer esta requisito. Con esto se ha establecido dichas latencias radio por debajo de 1 ms y de alta fiabilidad de hasta el 99,9999 % en algunos casos de uso.

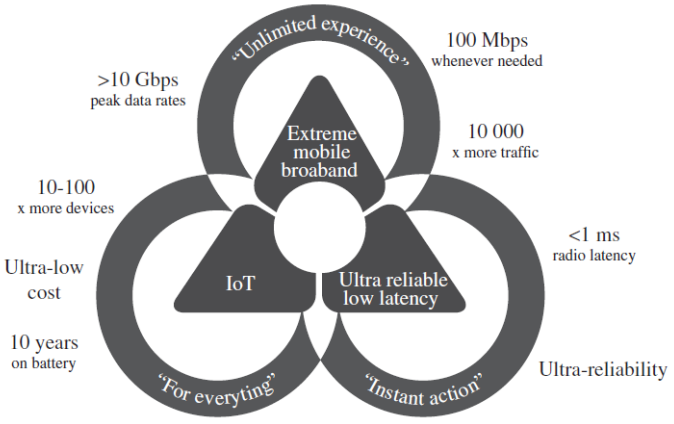


Figura 2.12: Objetivos principales de 5G NR [18]

En la Figura 2.12 pueden verse resumidos estos principales objetivos que 5G trata de conseguir para los diferentes casos de uso. Éstos suponen un gran reto dado que se trata de combinar tecnologías muy diversas donde algunas de ellas imponen estrictos requisitos temporales en cuanto al retardo, como es el caso de las comunicaciones URLLC, por lo que se ha de tener en cuenta una sincronización precisa entre todos estos dispositivos y la propia red.

Estos sistemas 5G requieren también de nuevas tecnologías que han ido surgiendo junto con las últimas versiones del estándar tales como *Massive Multiple Input Multiple Output (mMIMO)*, donde se emplean múltiples antenas en las estaciones base para aumentar la eficiencia espectral y energética del sistema. Esto se consigue focalizando los haces de las señales de radioacceso a través de un conformador o *beamformer* con el fin de reducir la potencia de transmisión gracias a la suma de las diferentes señales que llegan a través del canal con distintos retardos y con ello realizar un arreglo por interferometría para cada usuario de forma individualizada durante el *slot* temporal de transmisión en el enlace ascendente que le corresponda. En él ya se utilizan redes neuronales con pesos para acomodar la respuesta a las variaciones estadísticas de un canal *multipath* y las posibles interferencias por medio del error y el debido ajuste del retardo entre antenas para realizar un barrido con los mismos recursos espectrales entre los dispositivos. Gracias a esta técnica ya no es necesario el uso de *Cell-specific Reference Signal (CRS)* características de cada celda para la estimación del canal en los dispositivos puesto que en 5G se transmiten como la propia información, con lo que se reduce dicho consumo de potencia y con ello las interferencias ya que no se producen si no se transmiten datos de usuario. Además, son autoadaptativas en función del efecto Doppler, lo cual permite la movilidad a velocidades mayores, algo muy útil en los vehículos de conducción autónoma. En definitiva, eleva la directividad de dicho haz conforme se añaden más antenas permitiendo así un mayor alcance con un menor consumo de potencia aprovechando esa propagación por el mejor camino y aprovechando los mismos recursos entre los dispositivos.

Además, el *network slicing* ya mencionado anteriormente para la segmentación del tráfico en redes virtuales para diferentes casos de uso como puede ser el de mínimo retardo URLLC mediante un planificador radio ubicado en la estación base que selecciona dinámicamente los recursos espectrales y temporales para cada flujo en base al QoS establecido para cada. También se pueden tener en cuenta otros parámetros como la disponibilidad, la capacidad, la velocidad, etc.

El uso de bandas de frecuencia queda entre los 400 MHz y los 90 GHz, centrándose en comunicaciones de mayor alcance y de mayor velocidad, respectivamente. Con esto, 5G hace uso de canales de 100 MHz en la banda inferior a los 6 GHz y 400 MHz para superiores a ésta. Un esquema sobre las bandas de frecuencia empleadas por sistemas 5G puede verse a continuación con la Figura 2.13. Nótese que las ondas milimétricas o *mmWave*, aquellas por encima de los 52 GHz y hasta los 90 GHz ya mencionados, proporcionan mayores tasas de datos pero con un menor rango de cobertura, en torno a la centena de metros, por lo que su uso actual es más bien experimental[20]. Además, conforme se aumenta la frecuencia se reduce la potencia consumida dada la desaparición de buena parte de interferencias con otras estaciones base y entre dispositivos dado tal alcance en espacios cerrados y más pequeños. Por tanto, las frecuencias utilizadas así como los anchos de banda quedan muy por encima de la anterior generación LTE-Advanced o 4.5G, la cual solo podía agregar canales de 20 MHz para llegar al ancho inicial de 100 MHz de 5G y hasta 2.6 GHz, aumentando así considerablemente la eficiencia espectral al reducir los espaciados.

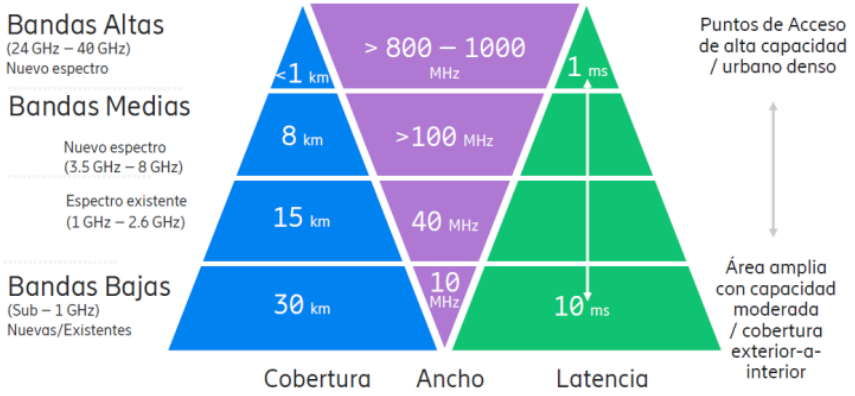


Figura 2.13: Bandas de frecuencia de sistemas 5G según su uso hasta los 40 GHz

Con ello, parece lógico pensar que los enlaces de comunicación para *slices* URLLC deben corresponder con un mayor ancho de banda a frecuencias más elevadas para distancias más cortas y así reducir ese retardo junto con el tiempo de transmisión de la información ya que se paraleliza este proceso entre símbolos *Orthogonal Frequency Division Multiplexing (OFDM)*. Es por este motivo por el que la duración de cada símbolo puede reducirse en *mini-slots* de hasta 0,065 ms frente al 1 ms de LTE, con lo que se minimiza este retardo de toda una subtrama de 14 símbolos en la RAN por debajo de 1 ms en la parte RAN, mientras que en LTE era de unos 7 ms. De esta forma, la duración de una trama completa, al componerse de 10 subtramas, será de 10 ms. Para comprender mejor la estructura de estas tramas, conviene visualizar la Figura 2.14.

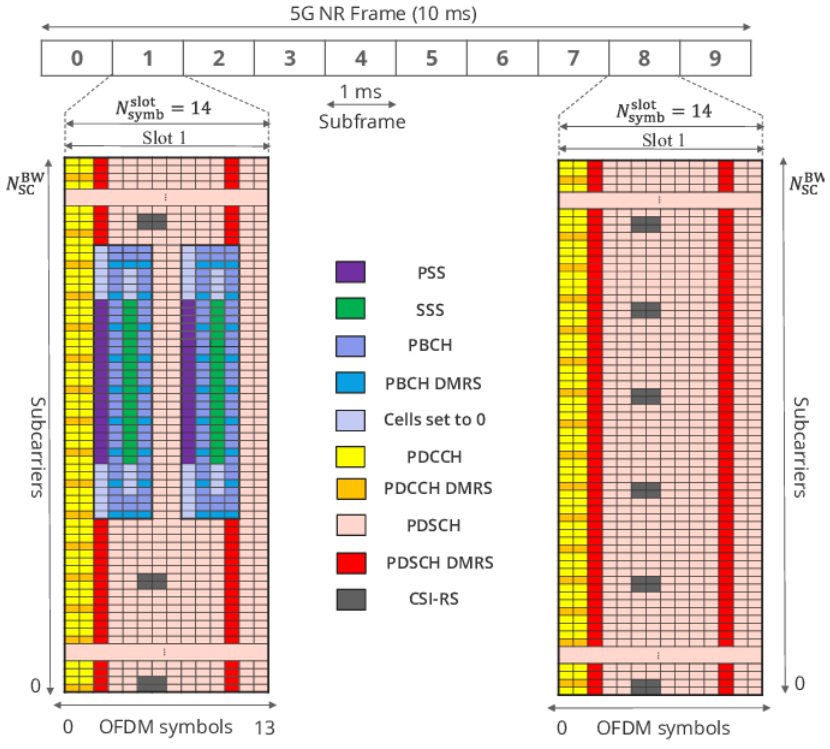


Figura 2.14: Estructura de las tramas en 5G NR en *downlink* [21]

Nótese que estas subtramas están formadas por símbolos y portadoras, formando así la unidad más básica de recurso radio conocida como *Resource Element (RE)*. El conjunto de 7 símbolos, lo que se denomina como *slot* con duraciones entre 0.125 ms y 0.5 ms, si son transmitidos a través de 12 subportadoras conforman un bloque de recursos o *Resource Block (RB)*. Por tanto, cuanto mayor sea el número de RBs, mayor será el ancho de banda utilizado. A través de estos recursos radio se comparte la información en los diferentes canales existentes entre el gNB y el *User Equipment (UE)*, tanto en el enlace ascendente como en el descendente. Entre dichos canales, se destacan a continuación los siguientes:

- ***Physical Downlink Shared Channel (PDSCH)***: Canal descendente utilizado para la transmisión de datos de usuario en un canal seguro autenticado.
- ***Physical Downlink Control Channel (PDCCH)***: Se emplea en el enlace descendente para la transmisión de la información de control con el fin de adaptarse a las variaciones del canal con *Hybrid Automatic Repeat reQuest (HARQ)*.
- ***Physical Random Access Channel (PRACH)***: Se utiliza en el enlace ascendente para establecer una llamada o transmitir una ráfaga de datos.
- ***Physical Uplink Shared Channel (PUSCH)***: Se emplea en el enlace ascendente para la transmisión de datos de usuario, reservándose así un canal seguro.
- ***Physical Uplink Control Channel (PUCCH)***: Se trata del canal destinado a la información de control en el enlace ascendente en el que se incluye la retroalimentación del estado del canal.

Adicionalmente, el uso de frecuencias más altas permite la flexibilidad en cuanto al número de dispositivos que se conectan a la red y la reducción en el tamaño de las antenas, por lo que es óptimo aplicar técnicas MIMO con el fin de aumentar la eficiencia espectral, reducir aún más la potencia de transmisión y lograr celdas de mayor superficie en caso de así necesitarlo. Por otro lado, los sistemas 5G también hacen uso de otras técnicas ya existentes en anteriores generaciones de redes móviles como el uso de algoritmos como HARQ asíncrono junto *Low Density Parity Check (LDPC)* para la detección y corrección de errores; las modulaciones QPSK y M-QAM; etc. Sin embargo, lo que realmente hace diferente a 5G respecto a las generaciones anteriores ha sido una perspectiva más orientada a la computación *Fog*, utilizada en la Industria 4.0. Ésta ha consistido en la centralización del procesamiento de la señal captada por las antenas de las diferentes estaciones base en una arquitectura de operaciones distribuidas mediante los interfaces ya comentados y en lo que se conoce como O-RAN para así ganar en interoperabilidad con los RU, DU y CU; y la distribución de las funciones del núcleo de la red 5G con el fin de disminuir así la latencia más allá de lo visto en la parte radio. Además, con el uso de la interfaz *eCPRI*, la cual está basada en Ethernet, se logra reducir el retardo con la división de todo este proceso que hasta ahora tenía ocasión directamente en la estación base. Dicho *split* se produce en el DU para la capa física tanto para el enlace ascendente como para el descendente, dividiéndola así en inferior y superior en diferentes alternativas desde el mapeo de los RE hasta las transformadas de Fourier, como se muestra en la Figura 2.15. El DU es por tanto el encargado de procesar las capas inferiores a este corte, con lo que se lleva la mayor parte de la computación al CU para lograr esta centralización del sistema con tasas de transmisión hasta 9 veces por encima del caso tradicional en el que ya se transmiten bits. Cuanto más bajo sea el *split*, menor será el retardo en la red de transporte, pero también

será mayor en ancho de banda necesario. De hecho, eCPRI reduce aún más el retardo dado que simplemente se encarga del muestreo de la señal en fase y cuadratura, pero como es el caso de la Industria 4.0 con un gran número de flujos producido por un mayor volumen de dispositivos interconectados y control se haría inviable dado el gran ancho de banda que requieren dichas muestras, sobre todo si se usan técnicas mMIMO.

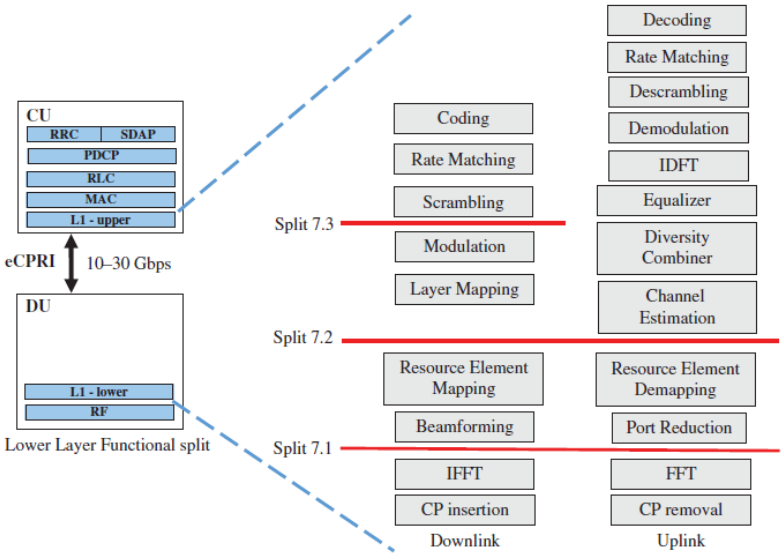


Figura 2.15: Alternativas de *split* capa 1 con eCPRI [18]

Nótese que este diagrama se corresponde con la integración de DU junto con RU en la propia estación base al tratarse de la arquitectura más extendida dentro de estos nuevos sistemas de radio 5G, aunque esta RU podría ser la encargada de controlar la transmisión y recepción de las señales, el muestreo, etc. En cualquier caso ambos acostumbran a tratarse de un DSP. Para ello debe conectarse al DU a través de la interfaz F2, pero este tipo de interfaz no es objeto de estudio en este proyecto. Además, el *split* 7.3 solo es posible en el enlace descendente, donde también se puede realizar desde la modulación del código. En la Figura 2.16 puede verse el resultado de interconectar estos elementos.

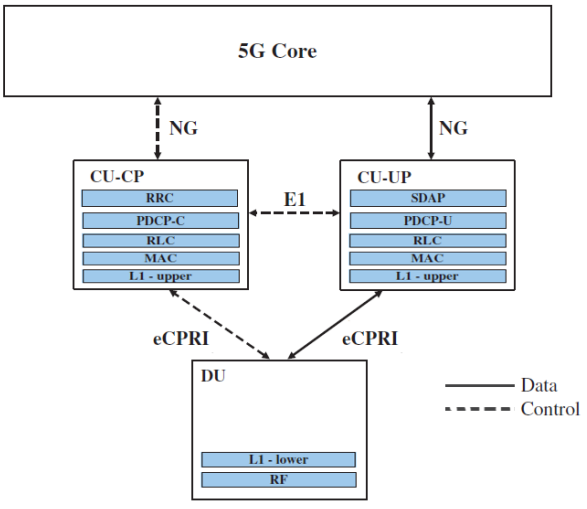


Figura 2.16: Arquitectura de unidades en *Xhaul* de 5G [18]

Con esto, el CU se divide en dos elementos lógicos: uno para el plano de datos (CU-UP) y otro para el plano de control (CU-CP) a fin de separar las funcionalidades de la capa *Service Data Adaptation Protocol (SDAP)* que trata de mapear la QoS de los distintos flujos o *slices* con las sesiones *Protocol Data Unit (PDU)* entre el usuario y la estación base frente a la capa de *Radio Resource Control (RRC)* más asociada a la asignación dinámica de estos recursos radio. Ambos se coordinan a través de la interfaz E1 para permitir la señalización entre ambos y operan sobre las capas *Packet Data Convergence Protocol (PDCP)*, MAC, *Radio Link Control (RLC)* y la parte superior de la capa física en esta unidad. También se conectan conjuntamente al núcleo de la red 5G a través de la intefaz NG, basada en IP para la convergencia entre tecnologías. Para ello, el módulo destinado al control está basado en el uso de *Stream Control Transmission Protocol (SCTP)* para comunicarse con el AMF mientras que el del plano de usuario lo hace con lo hace con *GPRS Tunneling Protocol - User Plane (GTP-U)* para intercambiar datos con el UPF, dos NFVs que veremos a continuación. Por tanto, las unidades CU serán las principales responsables de comunicar toda una región de estaciones base con la red troncal y de coordinar la transferencia de datos, la movilidad de los usuarios, la gestión de las sesiones, la planificación de los recursos radio, etc. Cada uno de estos CUs se conectará a múltiples DUs, por lo que su capacidad para procesar información es muy importante y, gracias a esa flexibilidad que permite la estructura O-RAN, podrá gestionarse a lo largo del tiempo de forma más eficiente y con un menor coste de operación y de capital. También podrá coordinarse con otros CUs a través de la interfaz Xn. Además, como se ha comentado anteriormente, eCPRI es la interfaz encargada de conectar los DUs con los CUs y está basado en Ethernet, por lo que el sistema permite la conmutación basada en paquetes y su sincronización con relojes, con lo que lo hace perfectamente integrable con redes deterministas como es el caso del sistema TSN estudiado en el anterior apartado, pues resulta bastante interesante permitir un bajo retardo también en esa red de transporte, la cual podría ser también parte de la red troncal TSN.

Para comprender mejor el funcionamiento de la red troncal de 5G conviene hacerlo a través de las NFVs. Éstas son las encargadas de materializar esa división entre el plano de control y el de datos de modo que la red gane en flexibilidad y simplicidad a la vez que se hace posible satisfacer las necesidades de cada aplicación por separado. En la Figura 2.17 se muestra la relación entre estos diferentes funciones.

A continuación se procede a realizar una breve explicación acerca de las funciones de las NFVs más relevantes para este proyecto dentro de una arquitectura basada en servicios y almacenamiento, lo que se conoce como *Service-Based Architecture (SBA)*:

- ***Access and Mobility-management Function (AMF)***. Parte del plano de control destinada a la señalización *Non-Access Stratum (NAS)* entre la RAN en la que se encuentra un UE y la red troncal de 5G para la autenticación, el registro en la red, la movilidad, el cifrado, favorecer el establecimiento de las sesiones, servicios de localización, etc. Por tanto, actúa como límite entre el núcleo de la red y la red de acceso inalámbrica, comunicándose así con los CUs a fin de supervisar las configuraciones necesarias capa a capa.
- ***Session Management Function (SMF)***. Se encarga del establecimiento y gestión de la sesión PDU entre la red y el UE, así como el mantenimiento del túnel entre el UPF seleccionado y la estación base, la asignación de una dirección IP a cada UE que se encuentre conectado a la red, etc. A éste se conectan por interfaces otros módulo para gestionar las políticas aplicadas, recolección de datos, etc.

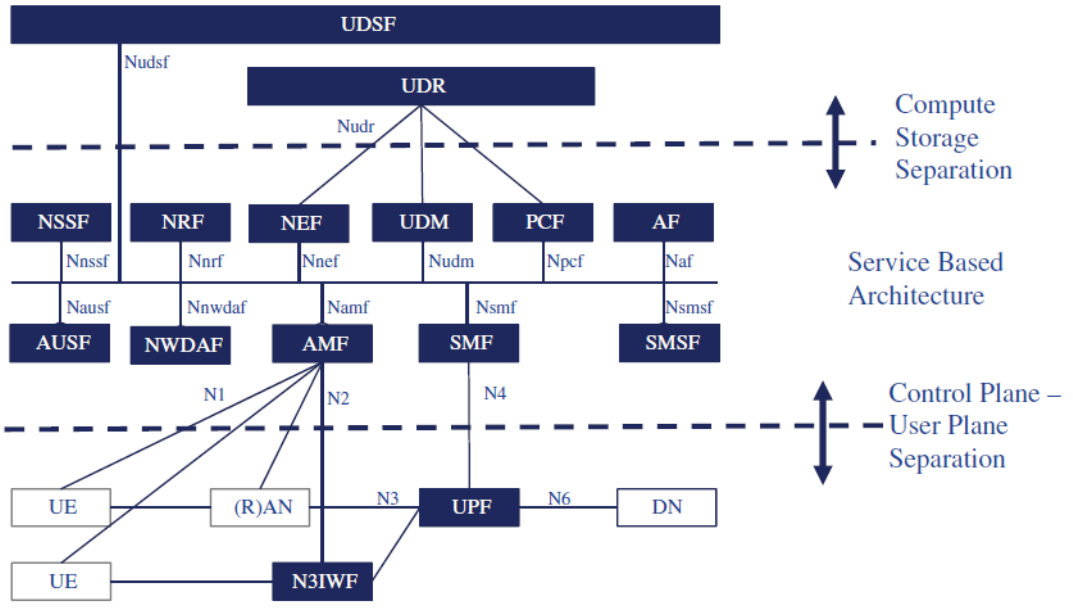


Figura 2.17: Funciones virtualizadas de red (NFV) de núcleo de red 5G [18]

- ***User Plane Function (UPF)***. Procesa el plano de datos del usuario tanto en enlace ascendente como descendente y lo encamina según la estación base correspondiente hacia la red troncal o externa. Por tanto, se encarga de su reenvío, el filtrado según las políticas aplicadas, la ejecución de QoS, los reportes de uso al SMF, etc. Con ello, participa en la asignación de los recursos en la red de transporte de 5G.

- ***Unified Data Repository (UDR)***. Es la base de datos común para todo tipo de estructuras estandarizadas (suscripción, políticas, aplicaciones, etc.). Solamente tres de estas pueden acceder y modificar el contenido del UDR: AMF, SMF y *Short Message Service Function (SMSF)*. El *Unified Data Management (UDM)* es el encargado de verificar procesos de autenticación para su acceso y con ello comprobar el estado de la información contenida en el UDR para el resto de interfaces. Por otro lado, cualquier otra función puede acceder a ésta para información no estructurada a través del *Unstructured Data Storage Function (UDSF)*, por lo que se puede almacenar cualquier aspecto relacionado con un UE desde el AMF.

- ***Network Exposure Function (NEF)***. Se encarga de gestionar los datos externos de la red, y con ello todas las aplicaciones externas que quieran acceder a los datos internos de la red 5G han de pasar por esta función a modo de *Application Programming Interface (API)* segura para cada utilidad.

- ***Policy Control Function (PCF)***. Es el encargado de aplicar las políticas de forma unificada mediante un *framework* común y realizar seguimiento del comportamiento de la red a fin de cumplirlas y garantizar el QoS a través de decisiones en base a la información contenida en el UDR.

- ***Network Slice Selection Function (NSSF)***. Asiste al AMF adecuado en la selección de los *network slices* asignados a los distintos usuarios de modo que cada uno de ellos se relacione con un identificador asociado al que le corresponda. Con ello, se coordina junto con el AMF para establecer esta asignación para cada uno.

- ***Application Function (AF)***. Realiza operaciones como el acceso a la función de exposición de red para recuperar recursos, la interacción con el PCF para el control de políticas, definir el encaminamiento del tráfico de las aplicaciones a través de los *Network Slice Subnet Instance (NSSI)*, la exposición de servicios a usuarios finales, etc. Por tanto, expone la capa de aplicación para interactuar con recursos de red 5G, tratándose así del punto de interconexión con otros sistemas con los que poder interaccionar, como podría ser el caso de TSN, a través de la exposición de los recursos de la red 5G a modo de descripción según las políticas.

De esta forma, el comportamiento de la red 5G es análogo al de cualquier esquema de red SDN, donde interviene un plano de datos de usuario pero también un plano de control a modo de *Fog* con mayor abstracción para adaptar el tráfico a las condiciones cambiantes del canal. Como venimos diciendo, cada aplicación requiere que la red 5G le proporcione una serie de recursos en función de sus necesidades para garantizar la QoS. En la Figura 2.18 puede observarse esta reserva de los recursos físicos en función del *network slice*.

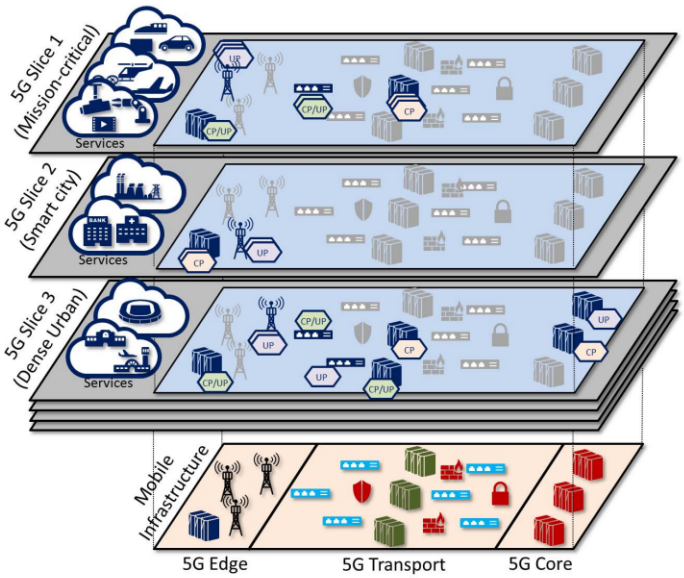


Figura 2.18: *Network slicing* con reserva de recursos en red 5G

Estos recursos se hacen más críticos en la parte radio al ser la que mayor retardo introduce, tratándose así de la modulación, la tasa de bits, el ancho de banda reservado a través de los RB, etc. Éstos vienen recogidos a través de un identificador denominado *5G QoS Indicator (5QI)*[22] y que los relaciona directamente con un determinado QoS para el flujo con valores de retardo máximo, el *Packet Error Rate*, etc. Este parámetro es normalmente reportado por el UE en función de su estimación del canal con la *Signal-to-Interference-plus-Noise Ratio (SINR)*, por lo que se trata de optimizar el uso de dichos recursos siempre y cuando sea posible; aunque también puede ser definido de forma dinámica por parte de la red según las políticas configuradas. Además, también depende del tipo de aplicación a la que preste servicio, estableciendo así el nivel de prioridad de cada uno. Por tanto, en el plano de control, el AMF junto con el NSSF serán los encargados de realizar tal gestión de los recursos, asignándole éstos a cada usuario en función del 5QI. Dichas asignaciones de recursos en función del 5QI, bien sea de forma individual o conjunta, forman la seg-

mentación lógica de la red a través de los *network slices* de modo que cada uno de ellos se comportan diferente a lo largo de la red.

Estos caminos *end-to-end* que cada flujo sigue dentro de la red de 5G son los que se denominan *Network Slice Instance (NSI)* y es, en definitiva, lo que caracteriza lógicamente el camino completo de un *network slice*. Éste se compone de diversas subinstancias NSSI a lo largo del recorrido por la red de transporte para formar dicho camino hasta su destino. Por tanto, un NSSI puede ser compartido por más de un NSI, aunque la reserva de recursos dependerá de los requisitos QoS de cada uno. De seleccionar qué *slice* corresponde a cada flujo se encarga la unidad NSSF vista anteriormente, aunque parte de la nomenclatura de la bibliografía consultada la denomina *Network Slice Management Function (NSMF)*. Sin embargo, el IETF[23] con la Rel. 17 propone nueva unidad llamada *Network Slice Subnet Management Function (NSSMF)* con el fin de gestionar esos NSSI en diferentes segmentos de esta red, así diferenciando entre la red de acceso (AN), la de transporte (TN) y la central (CN). También aparece la unidad *Communication Service Management Function (CSMF)*, la cual es responsable de traducir los requisitos de servicio del AF, con el que aplicaciones externas interactúan, a los requerimientos de *network slice* con los que trabaja el sistema 5G. Con todo ello, el CSMF solicita al NSMF una instancia acorde a dicho perfil de requisitos extremo a extremo para que éste decida asignando un NSI que los garantice a través de los NSSI que delega a las funciones RAN-NSSMF, TN-NSSMF y CN-NSSMF. De este modo, el NSMF realiza un mapeo entre los NSSI adquiridos por tales funciones con el NSI sobre el que trabaja a través de sus identificadores para encapsular los paquetes de datos recibidos en cada segmento de la red y mediante sesiones PDU para la comunicación extremo a extremo dentro de la red 5G.

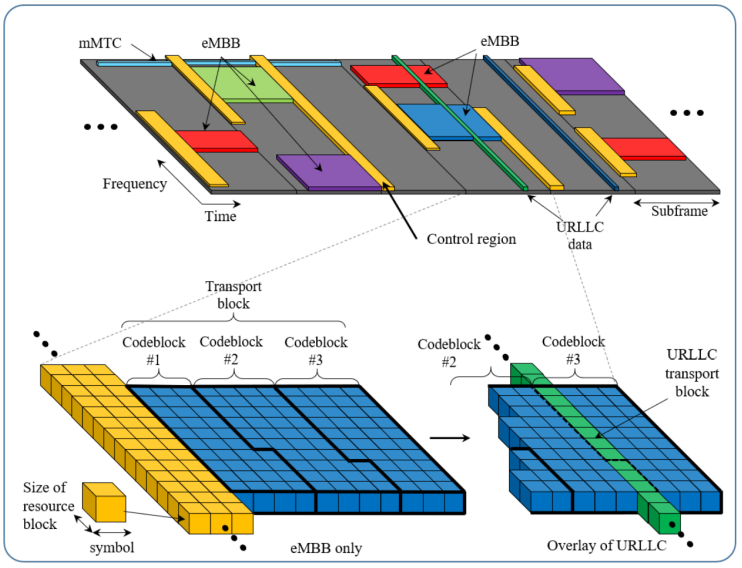


Figura 2.19: *Preemption* en red de acceso *downlink* 5G

Volviendo de nuevo a la parte de radioacceso, 5G permite lo que se denomina *puncturing scheduling*, interrumpiendo así desde el gNB la transmisión de flujos eMBB y mMTC para poder transmitir las comunicaciones URLLC, bien sean planificadas de forma periódica o esporádicas. Con ello, el sistema NR logra reducir el retardo en el acceso radio *downlink* multiplexado en TDMA, ya que *Frequency-Division Multiple Access (FDMA)* con un gran número de dispositivos como es el caso de la Industria 4.0 se hace inviable. Esto puede verse mejor con la Figura 2.19, donde dentro de la misma subtrama se utilizan los mismos

RBs para transmitir un flujo eMBB (azul) para luego focalizar sus recursos en un flujo URLLC (verde). En amarillo se presentan las señales de señalización de canal descendente de 5G o PDCCH. Nótese que estas transmisiones URLLC se producen en el menor tiempo posible como se ha comentado con anterioridad, es decir, la duración de escasos RBs (0,065 ms), aunque esto implica utilizar un gran ancho de banda para ello (véase Figura 2.13), pudiendo así afectar a múltiples comunicaciones. Esta transmisión multiplexada en el tiempo introduce un *overhead* adicional al comienzo y al final para señalizar al UE que a continuación recibirá el flujo de mayor prioridad que claramente afectará a la transmisión principal, en este caso la de eMBB, y que por ello deberá descartar esa parte recibida; aunque el de URLLC también se verá afectado por el *overhead* que le precede. Por tanto, la reserva continua de recursos para transmisiones URLLC lleva a un ligero desaprovechamiento de éstos, reduciéndose así el *throughput*. Sin embargo, para el caso de *uplink* se requiere la señalización por parte del UE hacia la estación base con el fin de poder transmitir un flujo URLLC, lo que lleva un mayor retardo. Una alternativa a ello es el llamado *grant-free*, el cual parte de una serie de recursos preasignados a un conjunto de UEs que transmiten en el *slice* de URLLC para que hagan uso de cierto canal siempre que requieran realizar una transmisión, aunque el uso compartido de éste puede dar lugar a colisiones; a menos que dicha transmisión esté sincronizada y planificada por ésta última, con lo que se podría ganar en velocidad y mejor aprovechamiento de los recursos.

2.4. Integración de 5G con TSN

Con esto, todo un sistema de radioacceso 5G como el descrito en el anterior Apartado 2.3 puede actuar como un *bridge* lógico en TSN como se trata en [5]. Si aplicamos esta idea sobre una red determinista en la Industria 4.0 tal y como se ha visto en el Apartado 2.2 con la Figura 2.11, se logra dotar de flexibilidad a la red de máquinas al mismo tiempo que permanecen conectadas a los controladores PLC sin necesidad de cableado.

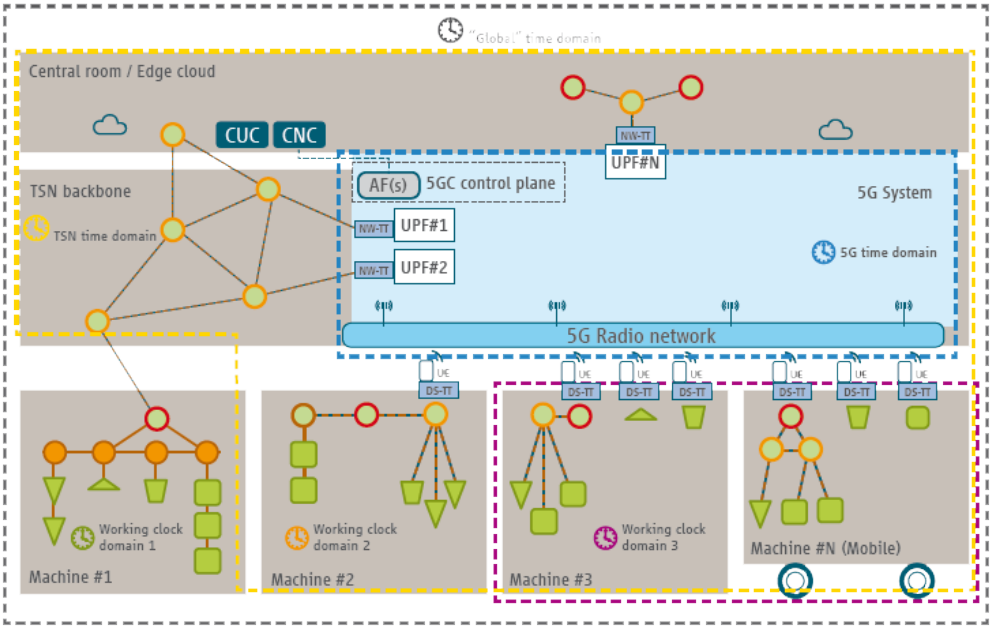


Figura 2.20: Red TSN completa en la Industria 4.0 con *Bridge* inalámbrico 5G [5]

Como se puede ver en la Figura 2.20, ahora toda la red 5G (azul) pasa a comportarse como una «caja negra» a la cual se enlazan otros nodos TSN para comunicar multitud de dispositivos que precisan de comunicaciones URLLC con los controladores PLC ubicados en otros puntos de la topología. Para ello, estos dispositivos se conectan con la red mediante un terminal móvil UE, y que bien podría ser único para cada uno y estar integrado dentro del mismo o bien ser común a un subconjunto de ellos que se encuentren próximos entre sí a través de un nodo TSN. En cualquier caso, el objetivo de ello es proporcionarle esa flexibilidad a partir de la conectividad 5G con este nodo inalámbrico de la misma forma en la que se ha explicado en el Apartado 2.3. Para ello, en lo que al plano de control respecta, el CNC de la red TSN debe comunicarse con el plano de control de la red 5G a través del AF con el fin de establecer en este *bridge* lógico de 5G los requisitos QoS de cada flujo y una configuración a partir de las capacidades que éste le proporcione, como ya se vio en el Apartado 2.2. Asimismo, el resto de NFVs realizarán internamente sus respectivos cometidos tal y como puede ser la reserva de recursos, el establecimiento de sesiones, *slicing*, revisión de las políticas, autenticación con los UE, etc. Entre ellas se encuentra también el UPF, única función del plano de datos empleada para el reenvío de la información de usuario hasta su destino con el resto de la red así como de la ejecución de las políticas y reportes de uso al plano de control de forma transparente a la red TSN. Sin embargo, en un *bridge* 5G, el UPF no se conecta únicamente con la red troncal 5G, sino que lo hace con otros *bridges* puramente TSN a través de enlaces físicos, así actuando como un nodo intermedio más. Del mismo modo, puede conectarse a otros de forma inalámbrica a través de UEs. Esto lo consigue a través de dos traductores TSN (TT)[24][25]: el *Network-side TSN Translator (NW-TT)*, ubicado junto a un único UPF; y el *Device-side TSN Translator (DS-TT)*, que se despliega de forma individual o como conjunto en el mismo UE. Ambos se utilizan como interfaces para la temporización de las tramas gPTP transmitidas periódicamente a través del *bridge* 5G, marcando así un tiempo de ingreso (TSi) y uno de egreso (TSe) con el reloj interno del sistema 5G sobre la misma trama de modo que se pueda controlar el tiempo de residencia en éste para comunicarlo con la corrección a los nodos TSN vecinos, siendo así el que probablemente más retardo introduzca por las condiciones cambiantes del canal. Nótese que estos tiempos se distribuirán ya por cada dominio de reloj de gPTP configurado, como puede verse en la Figura 2.21. Además, ambos traductores poseen funcionalidades a modo de PSFP para filtrar tráfico, *hold-forward* con dicha periodicidad de las tramas gPTP para reducir de forma sincronizada el *jitter* causado principalmente por la RAN, o el reporte de capacidades en capa 2 del sistema mediante LLDP para sintetizar así la topología, tal y como ya se comentó.

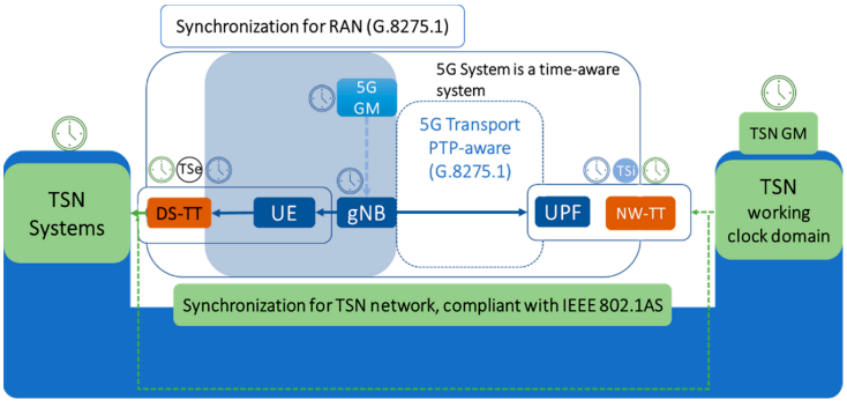


Figura 2.21: Esquema de dominios temporales en Bridge TSN de 5G [26]

Con todo ello, estos traductores así como los UE, la estación base gNB y los UPF forman el plano de datos completo de este *bridge* lógico. Dentro de este sistema pueden existir múltiples UE y sus respectivos DS-TT, cada uno actuando como un puerto virtual hacia los dispositivos u otros *bridges* TSN como ya se ha visto. Esto puede ocurrir de la misma forma con los UPF junto con los NW-TT, aunque éstos suelen conectar con nodos TSN troncales. Dado que solo se despliega un NW-TT por cada UPF, en el caso en el que se pretenda contar con varios puertos de entrada o salida a través de ellos es necesario configurar múltiples UPFs, de hecho ha de contarse con al menos uno por *slice* para diferenciar los distintos tipos de tráfico según su prioridad. En la Figura 2.22 se muestra un caso en el que se presenta esta situación, con lo que dentro de este *bridge* lógico se pueden establecer múltiples VLANs gracias a este sistema de *network slicing* a través de las sesiones PDU entre UE y UPF en los NSI ya estudiados. De tratarse este modelo como caja negra, bastaría con que la red TSN conozca los retardos producidos para cada *slice* entre estos puertos virtuales con el fin de llevar a cabo una planificación adecuada a sus características, abstrayéndose así de las rutas seguidas por cada sesión. Además, dado que TSN permite replicar las tramas a través de FRER, se pueden aprovechar los recursos de la red de transporte de 5G a través de dichos *slices* para cada una de las rutas seguidas por el mismo identificador a través de distintas sesiones PDU de modo que el sistema sea aún más robusto ante posibles fallos que puedan dar lugar a pérdidas, pues aquí lo más relevante es el NSI que establezca el NSSF para un QoS determinado a través de su identificador de modo que estos flujos encuentren siempre un camino alternativo para llegar a su destino con el menor retardo posible.

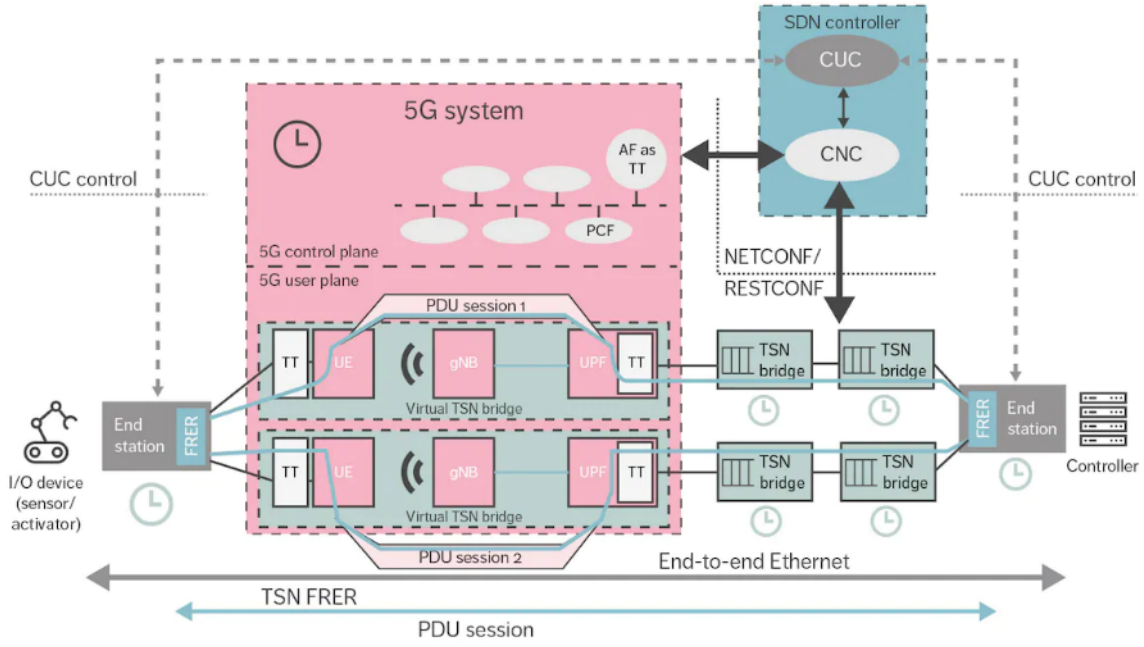


Figura 2.22: Sesiones PDU en *bridge* 5G integrado en red TSN [25]

Para que todo ello sea posible, la red de 5G debe realizar primero un proceso de preconfiguración con el fin de recabar información sobre sus propias capacidades a través del número de puertos virtuales formados por los UPFs y UEs junto con los respectivos traductores NW-TT y DS-TT con los que cuentan. Entre estas capacidades se encuentran principalmente el *throughput* del enlace que une a estos puertos virtuales con otros elementos de la red TSN y el retardo latente entre éstos por el tiempo de residencia mar-

cado en el plano de datos de la red de transporte 5G. También es importante realizar con ello un mapeo entre QoS del que se dispone tanto en la RAN (5QI) como la red de transporte (NSI/NSSI) con las distintas prioridades requeridas para permitir el *network slicing* dentro del propio *bridge* 5G, implementando así las distintas VLANs en las que se puede dividir. De este modo, el AF recogerá toda la información relacionada respecto a estas capacidades para cada una de ellas, la cual se almacena en el UDR por cada NFV responsable junto con su respectivo identificador; y proveerá al CNC de un informe para que éste actúe ejecutando las configuraciones acorde a las necesidades de las aplicaciones que hacen uso de la red. Sin embargo, el AF no expone toda la información ya que al CNC no le interesa conocer en detalle aspectos como la movilidad de estos puertos, por lo que se da a conocer únicamente la información relevante de forma estandarizada en base a LLDP. Entre estas configuraciones estarán la asignación de los tiempos a cada cola así como del híperperiodo, los reportes o las políticas PSFP que han de seguir los traductores TSN. En la Figura 2.23 puede verse esta segmentación del tráfico en diferentes QoS según su naturaleza a través de *network slicing*.

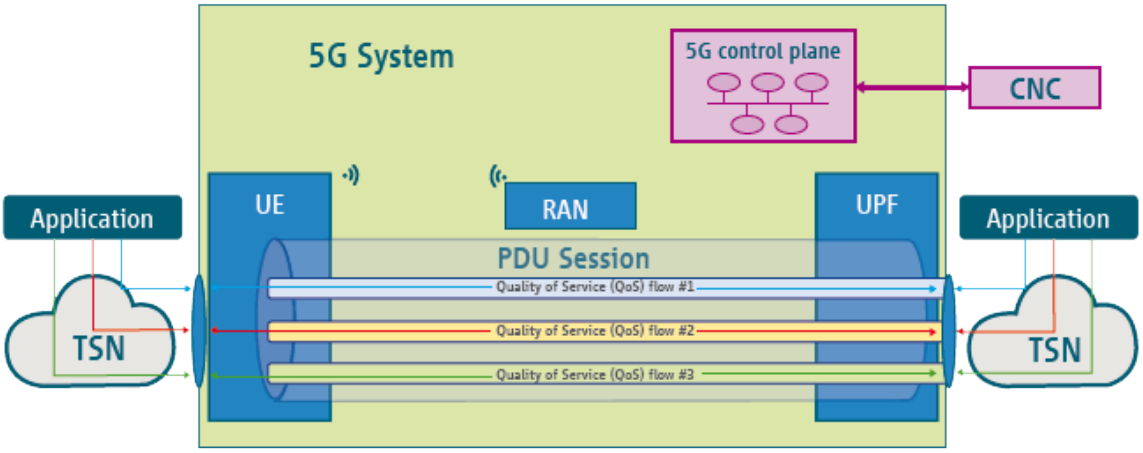


Figura 2.23: Mapeo de flujos con QoS *slicing* en *bridge* 5G [5]

Con esto, el AF recibe información sobre dicha planificación llevada a cabo por el CNC sobre el *bridge* 5G para que de esta forma las NFV correspondientes adapten sus parámetros a los diferentes flujos (e.g. políticas PSFP) y los nodos que conforman la red de transporte sean configurados de modo que, cuando el tráfico de TSN llegue a los puertos virtuales del *bridge* 5G, éstos sean capaces de identificar cada PCP con su correspondiente sesión PDU en un mismo perfil QoS. Dada la gran importancia en la sincronización, las tramas gPTP se transmiten a través de las VLANs de mayor prioridad a las que también pertenecen todas las comunicaciones críticas de control. También, cuanto más se reduzca este retardo, menor será en sí la magnitud del *jitter* y por tanto más se aproximará esta red de radioacceso 5G como *bridge* lógico al determinismo que se persigue con TSN. Con ello, ambos traductores podrán ajustar estos tiempos de retransmisión a través de *hold-forward* tal y como se consigue generalmente en un *bridge* TSN configurando los tiempos de transmisión de cada PCP, aunque en este caso no se trabaja con colas sino con *network slices* de 5G en sesiones PDU difererenciadas, por lo que se emula su funcionamiento siempre y cuando éste cuente con tales funcionalidades (clase A).

Respecto a esta comunicación entre ambas tecnologías, otras publicaciones, como es el caso del proyecto europeo 5G-SMART [27], se distinguen por dotar al AF de un módulo

exclusivo que provea tales mecanismos de gestión al CNC, éste es el TSN-AF, aunque conceptualmente se tratan de lo mismo. De esta forma, el CSMF será el encargado de interpretar al NSMF sus comunicaciones con el CNC para que éste los aplique según corresponda para cada segmento descrito de la red 5G mediante los NSSMF. Con ello, tal y como se plantea en este documento, el procedimiento para gestionar un *slice* quedaría de la forma que se ve en la Figura 2.24.

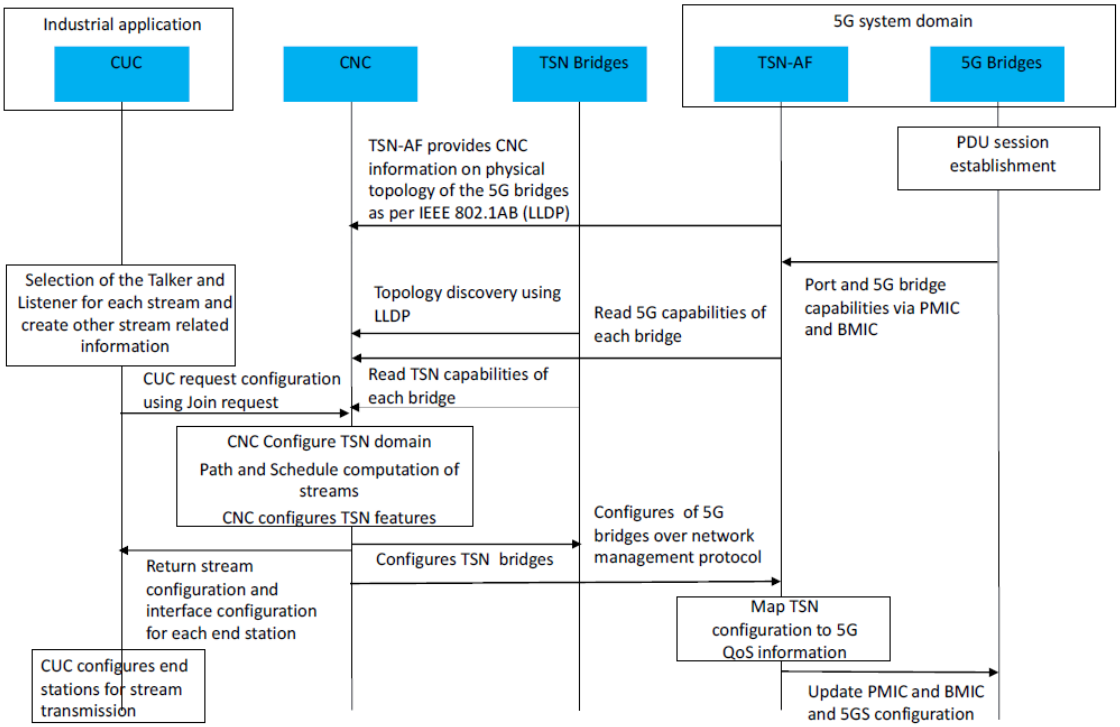


Figura 2.24: Procedimiento de configuración de *bridge* 5G en red TSN [27]

1. El UE correspondiente establece la sesión PDU con el propio *bridge* 5G de modo que éste intercomunica un extremo de la red con el otro a través de los segmentos de radioacceso y de transporte. Por cada *slice* habrá por tanto un DS-TT en un extremo y un UPF junto con un NW-TT en el otro. Estos extremos son realmente los puertos virtuales del *bridge* 5G de modo que en él se establecerán internamente unas rutas u otras con tal origen y destino para posteriormente ser conmutadas sus tramas a lo largo de la red de TSN hasta su destino. Con ello, realiza el cómputo sobre los retardos entre estos pares de puertos virtuales. Del mismo modo, aunque no es tan común, se comprueba esto mismo entre traductores del mismo tipo.

2. El TSN-AF provee al CNC de la información recogida mediante LLDP acerca de las capacidades de los nodos de la red de transporte así como de aspectos relacionados con su conectividad (la capacidad de los puertos DS-TT/NW-TT, el retardo entre ellos, el identificador VLAN, etc). Asimismo, el resto de nodos TSN también reportan sus capacidades al CNC a través de LLDP.

3. El CUC comunica al CNC los requisitos de estos flujos entre los dispositivos finales pertenecientes a una misma aplicación, identificando así grupos de *talkers* y *listeners* y sus respectivas características.

4. El CNC, en función de la topología física descubierta y sus capacidades, hace uso del estándar IEEE 802.1Qcc para realizar la configuración de planificación de tiempos para cada nodo del camino seleccionado para conectar ambos extremos. Esta planificación se realiza de acuerdo a los requisitos de estos flujos, dándole mayor prioridad a aquellos más críticos y haciendo uso de algoritmos de optimización para encontrar la mejor configuración posible que aporte un retardo y un *jitter* más acotados e incluso un mayor aprovechamiento del ancho de banda disponible. En el caso de que en el camino haya un *bridge* 5G, el CNC comunicará al TSN-AF información relevante para la configuración de los nodos pertenecientes a la red de transporte, así pasando a ser gestionados por el NSMF y, en concreto, por el TN-NSSMF.

5. El CNC comprueba que con tal configuración se satisfacen todos los requisitos y responde ahora al CUC acerca de la configuración realizada de modo que se le pueda notificar a los dispositivos finales con el fin de adaptarse a ella.

Otros trabajos relacionados como el de D. Ginthör et al. [28] también se han enfocado en visualizar el *bridge* 5G como «caja negra» para centrarse así en lo que sería un RAN-NSSMF, el cual es capaz de asignar los recursos radio de la estación base de modo que los tiempos de transmisión de cada *slice* así como las subportadoras OFDM utilizadas permitan establecer unos tiempos de residencia acordes a la planificación para que el retardo sea el esperado frente a las fluctuaciones ocasionadas por el propio canal.

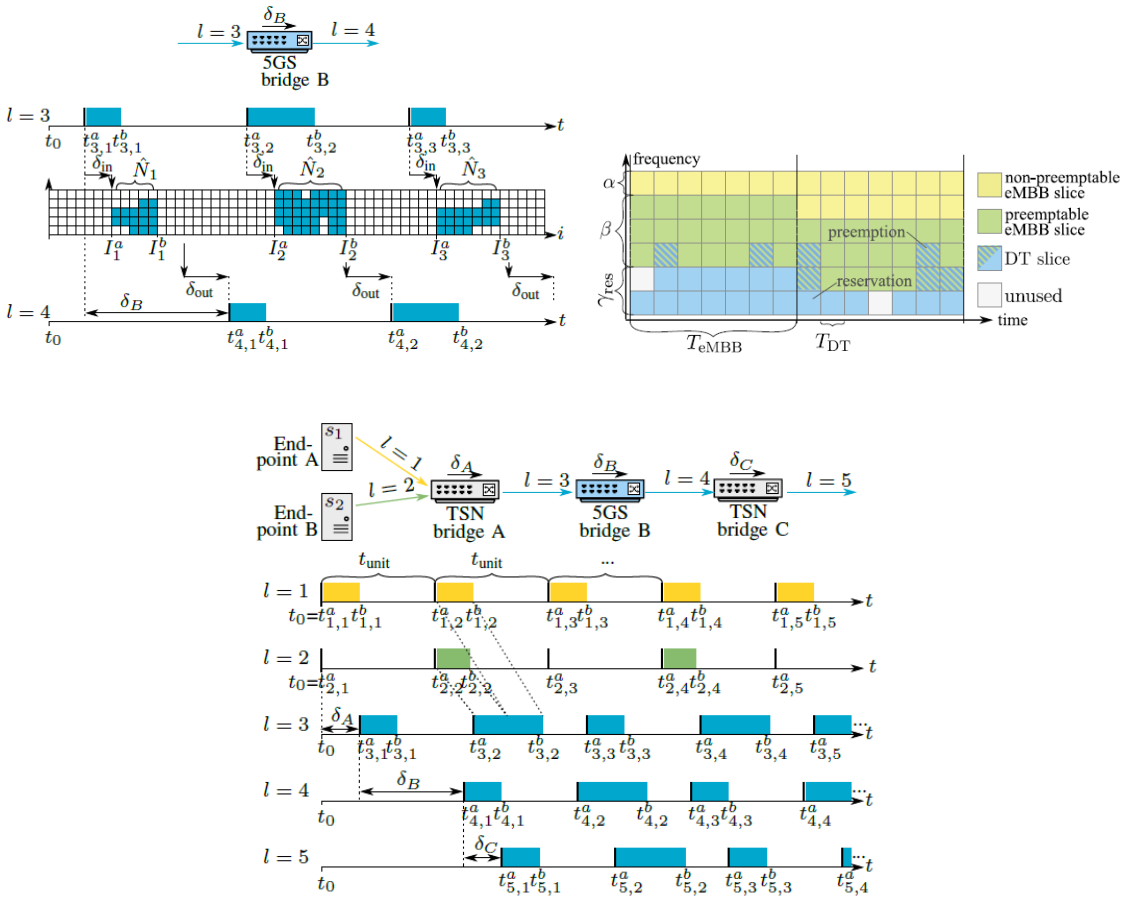


Figura 2.25: Adaptación de tiempos de trama (a) canal radio, (b) red TSN [28][29]

En otras publicaciones como la de D. Ginthör et al. [29] y la de Y. Zhang et al. [30] se contempla también el uso de *preemption* mediante el algoritmo *Earliest Deadline First (EDF)* con el fin de dar mayor prioridad al tráfico en el *slice* URLLC sobre cualquier otro para respetar estos tiempos cuando el acceso está más demandado, tal y como se explicó en el Apartado 2.3. Esta planificación se muestra en la Figura 2.25, donde se aprecia cómo en primera instancia el *bridge* 5G actúa de forma transparente en la red TSN con unos tiempos de transmisión «$t^{a,b}$» asignados a cada flujo y un retardo «δ» asociado al canal radio y a la propagación por la red de transporte, pero luego su comportamiento interno se basa en una asignación de portadoras OFDM a lo largo de los intervalos «I» discretizados por la duración de los RBs para respetar estos tiempos establecidos. Nótese que estos tiempos en la estación base se dividen en periodos dado que cada RB tiene una duración determinada. Por tanto, este sistema es modelable para que actúe en consecuencia a una situación del canal radio de forma optimizada a través de la asignación de dichos recursos «N» y su comunicación mediante el RRC. Con ello, los diferentes *slices* podrían ser reasignados con cierta flexibilidad de tal modo que el retardo por residencia en el *bridge* 5G sea el mínimo posible así como su *jitter* para el tráfico determinista (DT) pese a las condiciones cambiantes del canal y que el usuario reporta mediante el 5QI.

De este modo se puede llegar a mitigar el *jitter*, el cual puede dar lugar a desadaptaciones entre la llegada de las tramas a un nodo y los tiempos de transmisión y con ello desestabilizar el sistema de colas de prioridad. Sin embargo, pese a que este aspecto es muy importante en el proyecto de 6G-CHRONOS para dotar de mayor determinismo a la parte radio del bloque 5G y permita que la estación base actúe como uno nodo TSN más con unos tiempos de transmisión en sincronía con el resto de la red, este trabajo se abstrae de su implementación de momento de esta parte dada la complejidad de todo el sistema, suponiendo así que podría permanecer un *jitter* residual medible que se ha de tener en cuenta durante la planificación. De hecho, tal y como presentan también H. Zhou et al. [31], se podría modelar mediante técnicas de Inteligencia Artificial para casos más complejos. En él se emplean agentes inteligentes por cada *slice* de modo que éstos compiten por los recursos radio limitados para conseguir una solución que optimice el uso de éstos. Para ello se hace uso de la técnica de *Q-Learning*, la cual se basa en aprendizaje por refuerzo automático para dar con la combinación de parámetros de los *slices* de forma dinámica a través de recompensas (*q-values*) por acercarse a los principales objetivos mediante ensayo y error, en este caso pueden ser el de reducir el retardo y el *jitter*.

2.4.1. *Slicing* en redes de transporte TSN

Como se ha mencionado con anterioridad, cabe la idea de implementar toda una red TSN compatible con gPTP dentro de la propia red de transporte de 5G vista con la Figura 2.21 con el fin de disminuir aún más tanto el retardo como el *jitter* dentro de las VLANs implementadas sobre el *bridge*. Tal y como S. Bhattacharjee et al. proponen en [4], se debe tener en cuenta este reloj GM interno de 5G para la sincronización entre los *bridges* TSN que lo forman dado que la propagación de estas tramas debe ser acorde a los tiempos que se marcan entre ambos traductores. De este modo, la referencia de tiempo de los conmutadores internos podrá diferir respecto a los externos que forman el conjunto de la red TSN. Sin embargo, si se decide aprovechar un segmento de ésta última para formar tal red de transporte dentro del *bridge* 5G, se requerirá que éstos trabajen con dominios de reloj diferentes para los distintos tipos de VLAN, es decir, el primero obtenido a partir del GM de 5G y el otro a partir del GM de TSN.

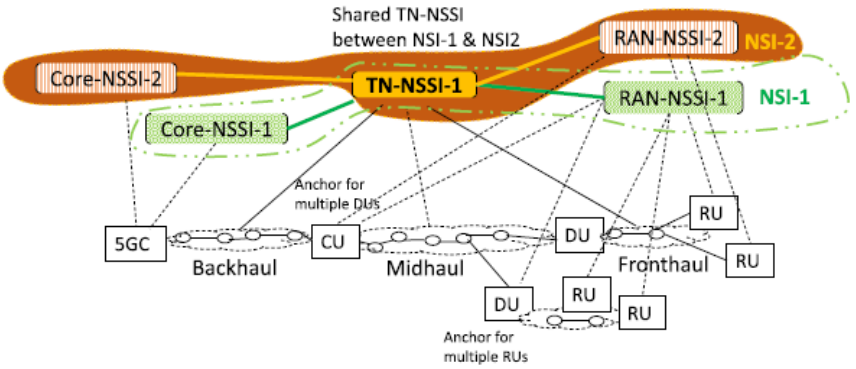


Figura 2.26: Red de transporte de slices en 5G [4]

Con esto se llegaría al principal objetivo del proyecto de 6G-CHRONOS, el de poder implementar *slicing* sobre redes de transporte 5G mediante nodos TSN de la manera en que se ve en la Figura 2.26. Como se puede ver, en esta red aparecen también las funciones distribuidas vistas en el Apartado 2.3 para el procesamiento de las tramas en sus diferentes capas. Asimismo, la topología se presenta como una división de segmentos *X-haul* entre dichas funciones, así con un *fronthaul* entre el RU y el DU, un *midhaul* entre el DU y el CU; y finalmente un *backhaul* entre el CU y el núcleo de la red en el que se encuentra el UPF. Esto implica por tanto que dichas funciones quedan distribuidas a lo largo de esta red para ganar en flexibilidad. Además, puesto que esta infraestructura es compartida entre los diferentes *slices*, se deben administrar los recursos a través de una planificación adecuada con los tiempos destinados a cada flujo para cumplir con los requisitos QoS.

En este mismo documento [4] se propone un *framework* capaz de realizar esto mismo mediante la implementación de un orquestador de la red generalizada de TSN. Hay que tener en cuenta que las mismas capacidades reportadas mediante LLDP por estos nodos TSN que forman la red de transporte deben ser también expuestas por parte del *bridge* 5G a través del AF, de modo que el CNC pueda ser capaz de gestionarlos de la misma forma a través de dicho orquestador. En esta arquitectura aparece la figura del *Multilayer Transport Network Slice Controller (MTNSC)*, el cual se comunica con el TN-NSSMF (Apartado 2.3) a través de la interfaz *tn-ml-nsi* y con el plano de control de la red TSN (Apartado 2.2) por la interfaz *tn-tsn-nsi*; siendo así el responsable de gestionar los requerimientos y políticas TSN de la red de transporte, las especificaciones de los NSSI en dicha red, su exposición de capacidades y monitorización, etc. En definitiva, se encarga de la gestión de los recursos asignados a los diferentes *slices* de modo que la red de transporte TSN pueda ser configurada de forma local en cooperación con la planificación llevada a cabo de forma centralizada en el CNC en función de los requisitos QoS. Este esquema puede verse en la Figura 2.27. Con todo ello, mediante el nuevo plano de control de TSN, se puede aislar un *slice* de otro con planificadores como el presentado en el caso síncrono de IEEE 802.1Qbv. De este modo, el orquestador debe tomar decisiones en cuanto a qué PCP asignar a cada *slice* así como un esquema de tiempos de transmisión a lo largo de la ruta más conveniente para ellos con el fin de minimizar los retardos. Esto debe hacerlo por tanto desde un punto de vista más general de la topología con el paradigma de control SDN. Con ello, las entradas que este módulo reciben son tales requerimientos y perfiles para cada *slice* como son el manejo, el filtrado y la agregación. Para ello, esta arquitectura trabaja sobre las primeras capas del modelo OSI (física, enlace, red y transporte) con el fin de soportar todas las funcionalidades locales de TSN pero también extremo a extremo con

el NSMF. Es por esto por lo que implementa bases de datos para guardar sus respectivos estados de los recursos reservados, así también en el *fronthaul* (RAN-NSSMF) y el *backhaul* (CN-NSSMF) dentro del mismo *bridge* 5G y con ello mapear el NSI con todos los NSSI de los diferentes segmentos y relacionarlos con los requisitos QoS de cada flujo. Todo esto se define mediante los requisitos de servicio de los usuarios de la red recogidos por el CUC y facilitados al CNC para ponerlos en conocimiento a través del AF del *bridge* 5G. Por tanto, a partir de ellos, en el *bridge* 5G pueden asignarse *slices* sobre los recursos dentro de la red de acceso y transporte de acuerdo a tales necesidades y con esto comunicarle al CNC tales características para que éste gestione los tiempos destinados a cada uno.

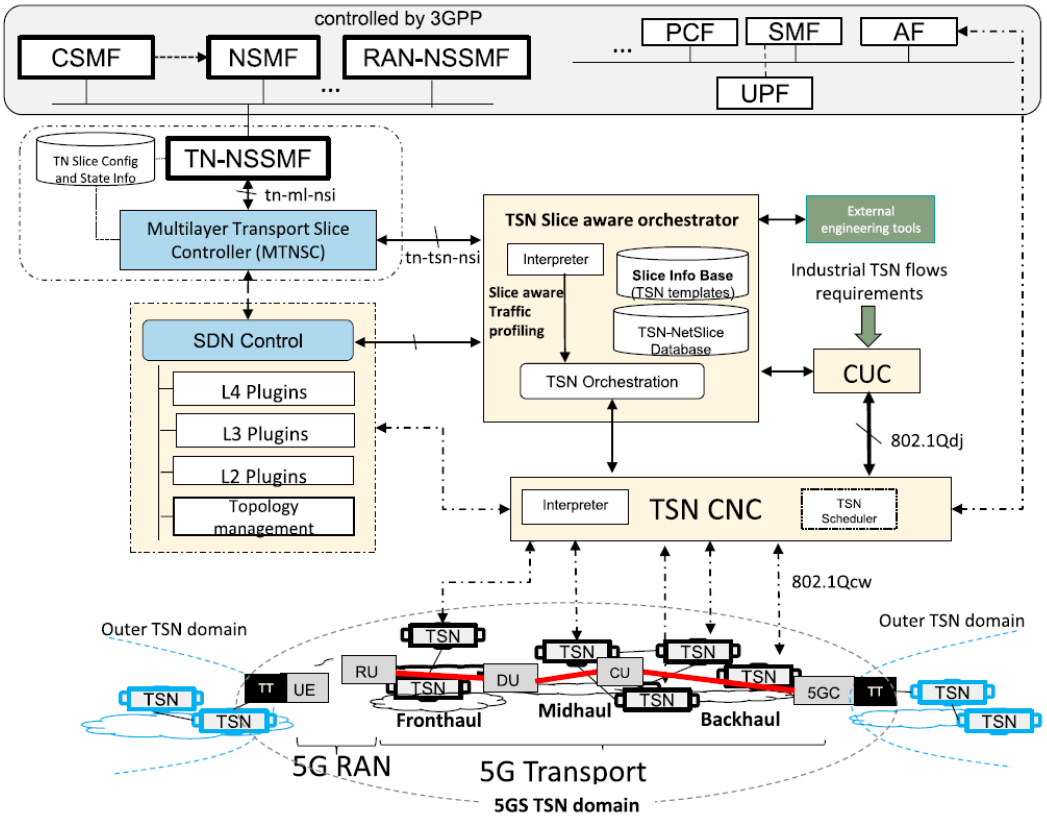


Figura 2.27: Arquitectura propuesta para *slicing* en *Xhaul* de TSN en 5G [4]

Además de todo esto, se propone dentro del mismo trabajo un módulo externo de herramientas de ingeniería sobre el plano de control de TSN, y con ello el orquestador estudiado, que permita resolver problemas de decisión en la planificación más complejos y de manera óptima a través de modelos de planificación avanzados. En este proyecto se propone el uso de Inteligencia Artificial como solución.

Sin embargo, dada la complejidad y el corto recorrido de estas soluciones con TSN como red de transporte, nos centraremos únicamente en el modelo de *bridge* 5G como caja negra en la red TSN, pues como se verá más adelante es posible extrapolar parte de sus resultados a este caso particular de red de transporte. De este modo, trabajaremos sobre una topología de red TSN orientada a la Industria 4.0 con un *bridge* virtual 5G como un nodo particular. Para ello conviene definir primero el problema a resolver como se verá a continuación con la Sección 3.

Capítulo 3

Definición del problema y soluciones propuestas

Si bien se ha realizado hasta ahora una revisión del Estado del Arte acerca de las tecnologías que, como es el caso de la integración de TSN y 5G, permiten la cuarta revolución industrial con la automatización de los procesos deterministas más críticos con una mayor flexibilidad, a partir de ahora se definen los problemas a resolver en este proyecto para que ésta sea posible ya que se pretende realizar un modelo de planificador TAS que permita las aplicaciones de la Industria 4.0 en sintonía con las redes inalámbricas. De este modo, tenemos definir por un lado la planificación de los tiempos asignados a cada flujo en la red de TSN y por el otro tenemos la integración de esta red con la de 5G.

3.1. Planificación de flujos en TSN

Para la planificación de flujos en TSN síncrono se debe contemplar el uso de la banda de guarda previa a la transmisión del tráfico determinista descrito anteriormente (DT) en el Apartado 2.2 sobre el resto de colas prioritarias y en especial el caso del *best-effort* (BE). Esta banda de guarda se emplea a fin de evitar que las transmisión de otras colas colisionen con los tiempos de transmisión reservados exclusivamente a estos flujos de mayor prioridad ya que, de producirse, sus tramas deberán aguardar en cola hasta que se complete dicha transmisión. Esto provocará por tanto que los tiempos designados a estos flujos no se lleguen a cumplir ya que en los siguientes saltos se encontrarán con la puerta de su correspondiente cola cerrada, aumentando así el retardo y desestabilizando todo el sistema ya que a esta cola llegarán nuevas tramas que no podrán ser transmitidas hasta que las anteriores lo hagan antes, pues se tratan de colas *First Input First Output (FIFO)*.

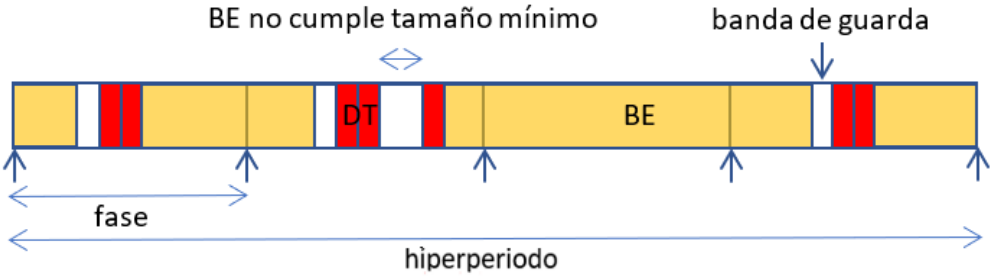


Figura 3.1: Asignación de recursos no optimizada

En la Figura 3.1 puede apreciarse que además, de no agruparse este tráfico adecuadamente, pese a cumplir correctamente con los requisitos de los flujos, se tiene como resultado un mal aprovechamiento de los recursos de red ya que aparecerán huecos entre estas transmisiones con un intervalo inferior al tiempo necesario para poder transmitir una trama característica de este tipo de tráfico, la cual puede llegar a adquirir como tamaño la MTU de 1542 Bytes vista en el Apartado 2.2. Por el contrario, estos huecos no afectarán al tráfico determinista ya que el GCL no permitirá la transmisión de otras colas durante este intervalo de tiempo, con lo que ninguna trama podrá colisionar. Esto se producirá en un entorno real debido a que existirán flujos que harán uso de la red con requisitos diferentes, comenzando por el origen y el destino, donde las tramas de cada uno llegarán a un mismo nodo en distintos instantes de tiempo; su priorización sobre cuál se ha de transmitir primero haciendo que el resto deban esperar en su respectiva cola; y los periodos con los que se transmiten, dando lugar a situaciones distintas por cada fase dentro del mismo híperperiodo. Esta problemática puede acentuarse aún más en un esquema de comunicaciones masivas M2M con requisitos aún más complejos fomentado por la aparición de los *bridges* virtuales 5G en la red TSN de la Industria 4.0 y donde se podría aplicar FRER para mayor redundancia en las transmisiones, dando lugar así a un menor aprovechamiento de estos recursos e impidiendo la convivencia entre diferentes tipos de tráfico para la que se pretendía diseñar esta red. En cambio, es posible revertir esto de modo que, manteniendo o incluso aumentando ínfimamente el retardo por el mero hecho de establecer otra ruta alternativa (IEEE 802.1Qca) diferente a la que intuitivamente nos puede dar como solución un algoritmo de camino más corto, siempre por debajo del tiempo establecido en los requisitos, se puede obtener como resultado un mayor nivel de utilización de los enlaces de la red TSN. Además, se busca que este algoritmo de planificación sea resiliente ante posibles fallos, de modo que pueda converger hacia otra configuración de tiempos tan pronto como sea posible sin llegar a duplicar el número de flujos en la red a través de FRER pues, como ya se ha indicado, puede provocar la aparición de estos huecos.

Por este motivo, resulta interesante conocer primero los algoritmos de planificación que podrían emplearse para el caso de TSN síncrono de modo que puedan ofrecernos un tiempo de cómputo razonable para unos buenos resultados en cuanto al retardo de estos flujos y de la utilización de los enlaces, pues se pretende optimizar ambos parámetros.

3.1.1. Métodos existentes de planificación

Dada la situación actual de la tecnología, existen diversos algoritmos para la planificación de los recursos en los nodos de TSN síncrono sobre los cuales no hay todavía un consenso sobre qué método puede ser mejor en cada caso como pueden ser los ya mencionados MILP o SMT, aunque también existen otros que optan por estrategias alternativas. A continuación se procede a comentar brevemente algunos de ellos.

Mixed-Integer Linear Programming (MILP)

MILP[32][33] se presenta como un algoritmo de optimización lineal en el que existe una función objetivo en forma de polinomio a minimizar por lo que, de existir una solución óptima en cuanto a retardos *end-to-end*, la encontrará. MILP se diferencia de ILP en que puede hacer uso tanto de números enteros como reales, lo que en este caso lleva a establecer una precisión con la que poder recorrer todos los valores en las combinaciones posibles de las variables que forman dicho polinomio. Para dicha función es necesario cumplir una

serie de condiciones: a) dado que Ethernet se basa en TDMA, dos o más flujos no pueden ser transmitidos de forma simultánea, sino que mientras uno se transmite el resto deben aguardar en cola; b) solo puede emplearse la planificación para un determinado flujo en los nodos que forman parte del camino de éste; c) el tiempo asignado a cada flujo en un nodo debe ser suficiente para su completa transmisión, contando con las fluctuaciones en el tiempo que puedan acontecer; d) el tiempo de transmisión en un nodo solamente puede comenzar una vez haya terminado de transmitirse en el anterior nodo, propagarse a través del medio y se haya conmutado hasta el puerto correspondiente en el actual nodo; e) idealmente, una trama debe esperar el mínimo tiempo posible en cola, por lo que la trama planificada para ir a continuación de la actual debe llegar sin anticiparse y sin demora, aunque basta con que llegue después de la anterior para respetar el sistema de colas FIFO, es decir, no llega antes que la trama planificada para transmitirse con anterioridad a ella; f) estos tiempos se reproducen de forma periódica según las características de los flujos, por lo que las anteriores restricciones se aplican; g) no existan bucles entre nodos. Esto puede traducirse a una serie de desigualdades de primer orden (lineales) las cuales acotan el espacio de búsqueda y que los tiempos asignados a cada flujo han de cumplir. Esto puede verse en la Figura 3.2 con los flujos *s* y *s'*.

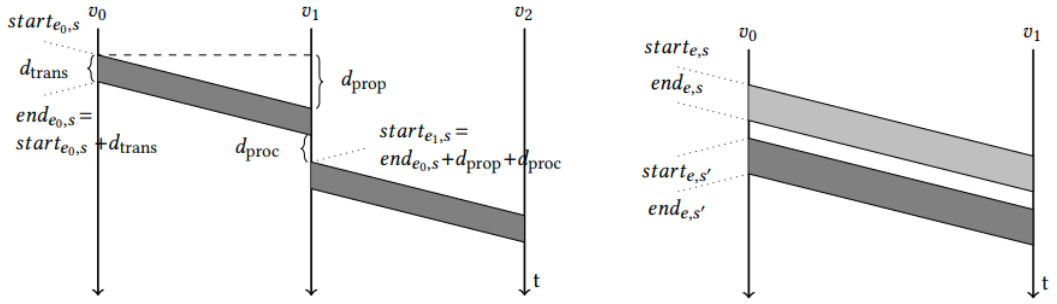


Figura 3.2: Esquema de planificación MILP [33]

Con todo ello, el algoritmo comienza a comprobar los resultados sobre el retardo de los flujos para cada combinación con el fin de hallar el mínimo. Sin embargo, el mayor problema que acarrea este sistema es el tiempo de cómputo que lleva explorar toda una serie de posibles combinaciones que asegure encontrar la solución óptima. Adicionalmente, la ruta seleccionada en este tipo de planificadores suele definirse como la más corta para cada uno de los flujos, por lo que en el caso de probar caminos diferentes que ofrezcan un mayor rendimiento, dado que se trata de un problema NP-duro al tratar de encontrar los valores en el tiempo que satisfagan dichas restricciones, el tiempo de cómputo crecerá considerablemente de forma exponencial, por lo que se debe recurrir a lo que se denomina «relajamiento de las restricciones» (e.g. discretización) y de los valores que puede recorrer el polinomio, lo cual probablemente lleve a configuraciones sub-óptimas. Asimismo, de encontrarse con un mayor volumen de flujos y/o con una topología más compleja, tal y como ya se ha mencionado, puede desembocar en una configuración lejos de ofrecer los mejores resultados en cuanto a retardo y aprovechamiento de los recursos. Además, si en ello incluimos la integración con 5G el problema se hace aún más complejo. Por último, la propia complejidad computacional de este algoritmo hace que tan solo pueda centrarse en la optimización de este retardo, por lo que puede dar lugar a una mala gestión de los recursos conforme se van añadiendo nuevos flujos. Es por este motivo por lo que es necesario encontrar un método de menor coste computacional con el que poder encontrar una solución optimizada de casos más complejos.

Satisfability Modulo Theories (SMT)

SMT[34] es un algoritmo muy similar al de MILP, donde también se deben satisfacer una serie de condiciones de primer orden para la planificación de los tiempos en los nodos que conforman la ruta de los flujos. Sin embargo, ya no existe una función para su optimización, sino que basta con que satisfagan tales restricciones de forma booleana. De este modo, se han de añadir restricciones en cuanto al retardo y/o cualquier otro parámetro a optimizar con el fin de cumplir tales garantías o incluso lograr que finalmente quede por debajo de un umbral de interés en lo que se conoce como *Optimization Modulo Theories (OMT)*, aunque de no ser posible jamás encontrará una solución y habrá que relajar dichas restricciones, esto es lo que se conoce como «problema indecidible». De nuevo, puede trabajarse tanto con valores enteros como reales, pero esto también implica un aumento en la complejidad del modelo y, por tanto, conllevará un notable incremento en el tiempo de computación. Por otro lado, este algoritmo puede proporcionarnos a priori si el problema puede ser o no resoluble dentro de las restricciones más básicas impuestas (e.g. retardo *end-to-end*), pasando posteriormente a mecanismos más orientados a la optimización. Por tanto, este método de planificación podría ser válido cuando no se pretenda lograr una configuración óptima, sino una sub-óptima que satisfaga con cierto margen los requisitos que puedan ser de nuestro interés. De esta forma, el problema pasaría a tratarse de un NP-completo.

Modelos heurísticos y metaheurísticos

Se conocen también modelos heurísticos como el presentado por M. Pahlevan et al. [35], donde se pretende lograr una planificación optimizada en cuanto a los huecos que existen entre transmisiones y que pueden provocar un uso menos eficiente de los recursos. Para ello ordena las tareas (o nodos) en función del coste que supone la comunicación entre éste y su predecesor según su gráfico de aplicación. Con esto, se realiza en cada nodo la planificación de cada flujo según un orden de prioridad a través de las tareas. De este modo, se requerirá computar primero los nodos por los que pasan previamente cada uno de los flujos a planificar antes (mayor prioridad) a fin de conocer cuáles son los tiempos de transmisión previos a éstos dada la interdependencia entre flujos, llevando así a un sistema recursivo entre tareas. Esto lo repite para cada posible camino entre el emisor y el receptor del flujo, comenzando desde el que ofrece la inserción más temprana, teniendo siempre en cuenta las restricciones ya vistas. En el caso de que los tiempos de un flujo violen el máximo retardo permitido, se pasa al siguiente mejor camino. De lo contrario, fija tal ruta con los correspondientes tiempos de transmisión en cada nodo. Este proceso lo repite para el resto de flujos, almacenando así los huecos que se produzcan entre tiempos de transmisión. Para minimizar esto, el algoritmo trata de hallar la solución que proporcione el menor tiempo de llegada a cada nodo. Tal y como muestra este trabajo, el algoritmo propuesto *Heuristic List Scheduler (HLS)* mejora considerablemente el hecho de usar únicamente el camino más corto *List Scheduler (LS)*, encontrando soluciones para los casos más complejos. Sin embargo, este modelo no contempla la minimización de los retardos, con la cual se puede conseguir una mayor capacidad de respuesta del sistema en su integración con 5G.

Modelos, como el propuesto por F. Dürr et al. [36], presentan la metaheurística para resolver la optimización de la planificación únicamente en cuanto a los huecos que se producen debido a las bandas de guarda a través del método de búsqueda Tabu, es decir, a través de la combinatoria de modo que se evite caer en máximos locales. Con esto establece

una analogía entre *Job-Shop Scheduling problem* y la restricción TDMA vista antes, en la que dos tramas Ethernet no pueden ser transmitidas a la vez a través del mismo puerto. Ya con todo ello y a partir de una solución inicial hallada a través de la heurística, como podría ser el método anterior, se irán seleccionando nuevas soluciones conforme se evalúe el uso de las bandas de guarda con el fin minimizarlo. Este algoritmo finalizará su ejecución cuando después de n iteraciones no haya logrado encontrar una solución con menor tiempo de guardas. El hecho de aumentar n puede favorecer la convergencia hacia máximos globales, aunque también conllevará un mayor tiempo de cómputo. Adicionalmente, este trabajo propone la compresión entre los tiempos de los diferentes flujos para reducir los huecos producidos entre transmisiones de tramas sensibles al tiempo, tal y como se ve en la Figura 3.3. De este modo, se podrá liberar más el espacio para que otros flujos también puedan ser transmitidos a través de ese enlace.

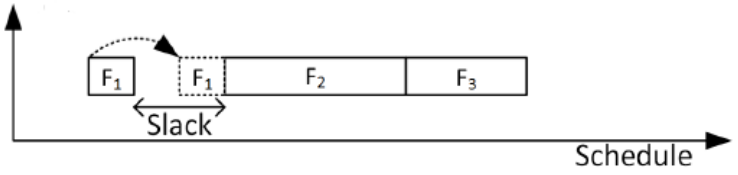


Figura 3.3: Esquema de planificación con compresión de flujos [36]

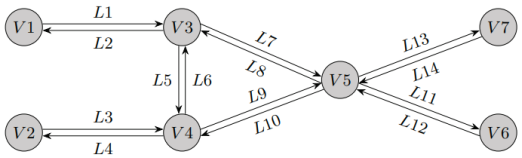
Sin embargo, esto provoca un aumento en el retardo por esperas en cola hasta tal punto que se puede llegar a incumplir las restricciones impuestas, a no ser que se replanifiquen los nodos previos en la ruta para retrasar la transmisión del mismo, lo cual puede llevar a tener que replanificar también la ruta completa de aquellos que se vean afectados, llevando a una cadena cuanto más compleja a medida que la topología crece y que elevará el tiempo de cómputo de forma exponencial al tratarse de un problema NP-duro. Es por este motivo por el que se cree oportuno considerar el retardo *end-to-end* en la optimización de estos parámetros. Además, de planificar una única cola para las comunicaciones críticas del *slice* URLLC, estos huecos podrían ser aprovechados por el tráfico de menor prioridad o *best-effort* siempre y cuando en ellos quepa la inserción de una banda de guarda previa a la transmisión del siguiente flujo URLLC a fin de protegerlo ante colisiones. En cambio, esto implica que habrá que insertar tantas bandas de guarda como huecos se produzcan, por lo que se tratará de minimizar igualmente este número de huecos.

Otros trabajos como el de F. Heilmann et al. [37] presentan modelos metaheurísticos con restricciones análogas y en los que se contempla la utilización de dichos huecos, aunque ahora se pretende segmentar aún más el tráfico dividiendo una misma cola de prioridad de TSN en otras dos a las que van a parar las tramas según cierto umbral de tamaño, es decir, pasan a formar parte de una de las 16 colas según su prioridad pero también su tamaño en Bytes. De esta forma, pueden recalcularse dichos umbrales para ofrecer una solución en la que se minimicen estos huecos. Puesto que el uso de estos huecos implica también transmitir antes las tramas contenidas en estas colas, se logra reducir además el retardo de éstas. Con ello, se podrán planificar los tiempos necesarios para cada cola de modo que se aprovechen mejor los recursos temporales dado que se conocerán a priori los tamaños de cada una. El principal problema de ello es que este modelo viola el paradigma general de las redes TSN en las cuales se plantea que un total de ocho clases debe ser suficiente para priorizar el tráfico, aunque se podría considerar una versión de cuatro clases divididas en dos colas cada una según un umbral de tamaño. De este modo, puede resultar fructuoso en

el caso de emplear más de una cola para tráfico de menor prioridad y con tamaños de trama predeterminados, pero no cuando únicamente se tenga una cola destinada comunicaciones URLLC y otra para *best-effort* que suele ser lo más común, por lo que se ha de encontrar un esquema que resuelva este tipo de problema de forma más genérica.

Modelos de reparabilidad

Existen modelos como el planteado por F. Pozo et al. [38] en el que se reencamina el tráfico ante cualquier posible fallo, como pueden ser la caída de un nodo o de un enlace. Dado que se tratan de comunicaciones de alta fiabilidad y mínimo retardo como es el caso del *slice* de URLLC en 5G, es muy importante contar con un algoritmo capaz de responder ante cualquiera de estas situaciones desfavorables en el menor tiempo posible. En este trabajo se propone un algoritmo basado en cualquiera de los anteriores modelos de planificación (e.g. MILP), pero teniendo esta vez en cuenta rutas alternativas para cada uno de los flujos en cada uno de los nodos. Con esto, distingue entre la posibilidad de planificar un determinado flujo y la de replanificarlo si uno de los nodos o los enlaces en el camino hacia su destino ha fallado. Sin embargo, es probable que afecte a más de un flujo y sus correspondientes requisitos QoS, por lo que requiere conocer los tiempos de transmisión de todos aquellos que se transmiten a la parte que ya no participa de forma activa en la red. Esto lleva a que debe descubrirse una ruta alternativa entre ambos extremos que atraviese la red TSN síncrona que cumpla con tales requisitos. Para ello, en una primera fase se comprueba que exista la posibilidad de que, a partir de los mismos nodos afectados por el enlace caído, se puedan desviar todos éstos a otros nodos intermedios que conecten a ambos sin afectar a los tiempos del resto de flujos ni a la planificación de los tiempos en saltos posteriores, esto es a través de todos los enlaces alternativos que parten de dichos nodos. Por este motivo será necesario que el camino encontrado entre ambos nodos sea el de menor número de saltos posible. En el caso de que no pueda realizarse esto, será necesaria una segunda fase, donde se tendrán que replanificar todos ellos en función de su prioridad, pudiendo así los desviados retrasar algunos de ellos únicamente en dicho enlace. Evidentemente, ésta llevará un mayor tiempo de computación, con lo que la respuesta al fallo será más lenta. En el caso de que, dadas las restricciones, esto tampoco sea posible, el fallo no podrá repararse. Un ejemplo de ello puede verse con la Figura 3.4 donde, dadas las restricciones, los flujos 1 y 2 no podrán planificarse en el desvío por el enlace 5 en el caso de que falle el enlace 7 dado que para ello habría que retrasar la transmisión de todos ellos en los enlaces 11 y 13.



Frame ID	Sender	Receivers	Tree Path	Period	Deadline
1	V1	V6	L1-L7-L11	8	8
2	V1	V6	L1-L7-L11	8	8
3	V2	V6	L3-L9-L11	8	8
4	V2	V6-V7	L3-L9-L11-L13	8	8

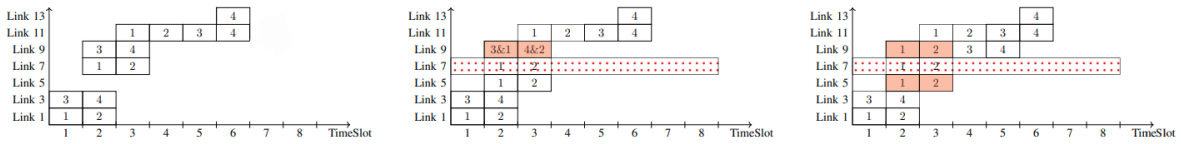


Figura 3.4: Esquema de replanificación en el desvío (a) previo al fallo, (b) fase 1 tras fallo en enlace 7, (c) fase 2 tras fallo en enlace 7 [38]

Es por ello por lo que esto tampoco será realizable en el caso de que la retransmisión de una trama en un nodo vaya justo a continuación de su recepción tal y como se vio en la Figura 3.2. Habría por tanto que maximizar el tiempo de espera en cola para los flujos 3 y 4 tal y como se propone en esta publicación, lo que conllevaría un aumento en el retardo. Además, esto sería aún más complejo en el caso de tener una topología *half-duplex*. Por tanto, en el caso de que todo esto no fuera realizable, habrá que retroceder a nodos anteriores a éstos para desviarlos por otras nuevas rutas que no pasen por dichos nodos, como también será en el que el problema se haya producido por que uno de estos nodos no esté activo. En resumen, habrá que volver a planificar la red TSN ante esta nueva situación. Para este caso quizá sería más conveniente definir a priori de forma *offline* la planificación de una ruta alternativa de modo que los conmutadores TSN puedan redirigirlo por otro camino tan pronto como detecten el fallo, y que en los siguientes saltos conozcan también con anticipo los tiempos a transmitir dicha trama respetando los ya planificados en cada enlace, aunque esto conlleva claramente a un alto coste computacional.

Por otro lado, la solución FRER podría paliar este inconveniente de forma permanente, aunque esto implica también una mayor carga a la red TSN y con ello mayor complejidad en la planificación ya que un mismo flujo ha de atravesar la red siempre por caminos diferentes, lo cual se aleja de optimizar los resultados como se viene pretendiendo.

Técnicas de Inteligencia Artificial

En este proyecto se pretende hacer uso de técnicas de *Machine Learning* para la optimización de la planificación en el segmento de la red de transporte TSN dentro del mismo *bridge* 5G así como en la red TSN que lo engloba. El mayor problema al que se enfrenta un algoritmo de automatización de la planificación es que trabaja con un problema NP-duro, lo que lleva a un coste computacional muy alto y obliga a que dicha planificación se haga *offline*, es decir, previamente a la puesta en marcha de la red con todos los escenarios posibles, como puede ser el caso de que un enlace o un nodo caigan, o también si se añade o elimina algún flujo con el fin de adecuar los tiempos al nuevo escenario. Sin embargo, esto también nos permite una más rápida convergencia ya que basta con buscar el estado en una lista ya clasificada. Por ello, para este tipo de problemas en los que se pretende optimizar el resultado, se ha concluido que dos técnicas adecuadas por que las se puede optar son bien *Reinforcement Learning*; o bien por Algoritmos Genéticos.

Con el trabajo de C. Zhong et al. [39] se nos presenta un *framework* novel conocido como *Deep Reinforcement Learning-based Scheduler (DRLS)* capaz de optimizar esta reserva para cada flujo de los recursos de la red de transporte así como de autoadaptarse a diferentes topologías, por lo que en caso de cualquier fallo, mediante la entrada del estado de la red a una red neuronal en lo que se conoce como *Deep Reinforcement Learning (DRL)*, pueda converger rápidamente sobre cuál es la solución adecuada para cumplir con todas las restricciones que se impongan. Esto implica que el algoritmo probará durante su entrenamiento configuraciones a través de ensayo-error hasta dar con una que cumpla una serie de restricciones, por lo que el objetivo de estas técnicas se centran más en satisfacer éstas que en optimizar el resultado. Algo muy importante en este trabajo es el uso del método de *Low Degree (LD)*, el cual permite asignar las tramas del tráfico de la red TSN sobre los recursos de ésta en cada puerto de cada nodo. Dado el gran espacio de búsqueda que esto puede suponer por la heterogeneidad de los tiempos con los que transmite cada dispositivo, se divide la asignación de tiempos del sistema en lo que se conocen como ranuras o *slots* de tiempo y cuyo valor es el de la máxima unidad de tiempo con la que se

transmite una trama o MTU de Ethernet. A partir de todo esto, en el caso del *framework TTDeep* presentado por H. Jia et al. [40] los flujos también se transmiten en la red TSN entre un número de fases resultado de dividir el híperperiodo, hallado a partir del MCM de todos los periodos de transmisión, entre el menor periodo con el que lo hace o el MCD. Con ello se define un *offset* o número de *slot* que indica la posición relativa dentro de la misma fase para cumplir con los tiempos establecidos. De este modo, un mismo *slot* perteneciente a diferentes fases puede ser aprovechado por distintos flujos que tengan frecuencias inferiores a la que establece el híperperiodo, y que por tanto dicho *slot* tenga calculado un «alto grado» de uso, como es el caso del *slot n* presentado en la Figura 3.5. Con ello, los de «alto grado» pueden contener tramas pertenecientes a flujos con mayor o menor periodo, mientras que las de «bajo grado» solo pueden contener aquellas de mayor periodo dado que otros flujos previos ya habrán ocupado las ranuras equivalentes en otras fases, impidiendo así la periodicidad de sus transmisiones. Además, el sistema debe ser capaz de seleccionar un *offset* adecuado de forma que reduzca el retardo de residencia así como el valor extremo a extremo con todos los saltos, principalmente para aquellas clases de mayor prioridad cuyos tiempos son más sensibles al tiempo, realizándose así este procedimiento en cada nodo de la red TSN según el tráfico que le llegue y en el orden que siguen las clases.

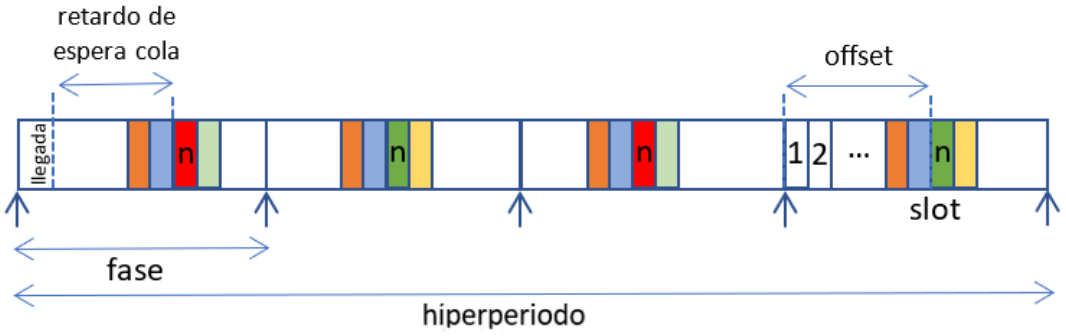


Figura 3.5: Asignación de tiempos con *Low Degree*

Sin embargo, el problema de discretizar el sistema en *slots* es que se pierde flexibilidad para trabajar con los tiempos, pues es perjudicial que el tamaño de estas ranuras esté fijado al tamaño de la MTU ya que, de existir flujos con tamaños de trama inferiores a dicho valor, se estará desaprovechando el tiempo restante hasta la siguiente ranura. Del mismo modo, también se desaprovechan estos recursos de tiempo en los enlaces al tratar las bandas de guarda ya que no pueden ser utilizadas. Además, todo esto se hace más complejo en el caso de que la transmisión se realice a modo de ráfaga, es decir, se envíen varias tramas de forma consecutiva en un mismo flujo, por lo que afectará a los grados de cada *slot* en dicha asignación. Esto también implicará un mayor tiempo de espera en cola de otros flujos, lo que a su vez se traduce en un mayor retardo de todos aquellos que vengan a continuación, así provocando una mayor dispersión en la planificación entre nodos en la que aparecen huecos donde no cabe la transmisión de tramas *best-effort*.

De este modo, en un sistema de aprendizaje por refuerzo se observa en cada instante t el estado de la red TSN para obtener información sobre la topología junto con la configuración de los tiempos así como de los requisitos de los flujos y con ello codificarla en vectores de estado. Tras ello, el agente realiza una acción sobre la configuración local o general de la red que modifique su funcionamiento para con ello obtener lo que se conoce como

recompensa, es decir, el cambio que produce en la evaluación del entorno tal modificación, como es el caso de las restricciones impuestas para cada flujo; no sin antes propagar el vector de cada flujo por la red neuronal para que ésta aprenda sobre los estados y acciones llevadas a cabo. Esta recompensa puede bien ser positiva o negativa, según las políticas de interés (e.g. que el retardo quede por debajo del máximo requerido), lo que se valora a través de los *q-values*. Un esquema de este modelo puede verse en la Figura 3.6. Nótese que es el conjunto de políticas las que definen la siguiente configuración a probar a partir de la recompensa del entorno, por lo que han de respetarse las restricciones del modelo. Sin embargo, ambas publicaciones contemplan un escenario en el que la red va aprendiendo automáticamente para acercarse a una solución de forma *online* mediante DRL, por lo que las fluctuaciones más dinámicas del tráfico debidas a una combinación no adecuada pueden alterar el resultado dada la dependencia entre los estados, además de que la medición a priori de las recompensas es muy compleja (e.g. parámetros que intervienen, pesos de éstos, etc.) y por tanto no puede predecir situaciones futuras. Esto lo hace más vulnerable a los posibles fallos que se puedan producir.

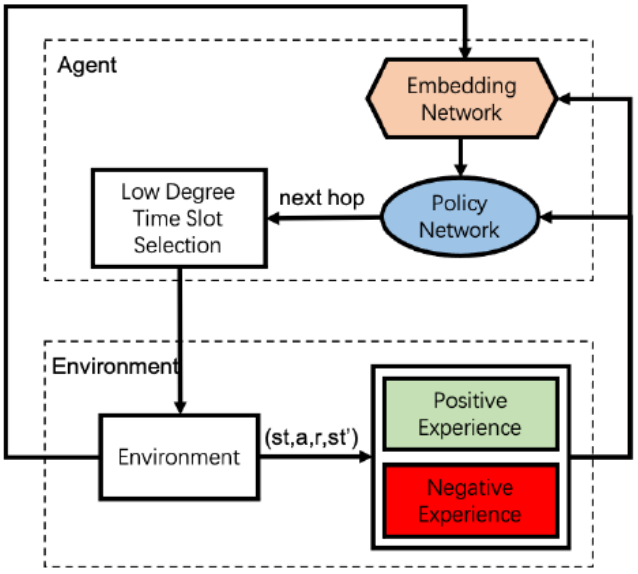


Figura 3.6: Esquema del modelo *Deep Reinforcement Learning Scheduler* [39]

Por otro lado, con publicaciones como la de F. Petroski et al. [41] se tratan los algoritmos genéticos como una gran alternativa a estos casos al tratarse de *frameworks* más simples en los que igualmente se basan en un aprendizaje que busca converger a una solución óptima sin recurrir a algoritmos de descenso del gradiente ni recompensas, siendo así menos costosos computacionalmente. De hecho, siguiendo esta línea de la metaheurística basada en los algoritmos genéticos, nos encontramos con el trabajo realizado por M. Pahlevan et al. [42] sobre la planificación en redes TSN a través de dicho tipo de algoritmos evolutivos. Se compara su rendimiento con el método heurístico LS para la planificación tal y como se vio en la anterior publicación [35], solo que esta vez se exploran rutas alternativas mediante algoritmos genéticos con el fin de minimizar estos huecos. Sin embargo, este algoritmo trata de minimizar los huecos a través de la minimización del retardo *end-to-end*, lo que puede llevar a que en situaciones más complejas como un número de flujos con distintas rutas, diferentes periodos, etc.; esto no sea así. Es por ello por lo que sus resultados muestran que tan solo es capaz de planificar 29 flujos en tales condiciones.

Los algoritmos genéticos no son de por sí una técnica de *Machine Learning* ya que su principal labor se enfoca en la optimización metaheurística a través de la combinatoria, pero son la vía más utilizada en este tipo de problemas frente a RL ya que basta con un modelo sencillo de aprendizaje automático que sea capaz de clasificar sus resultados, como puede ser un árbol de decisión o incluso una red neuronal a posteriori. En estos casos se trata de aprendizaje supervisado ya que el sistema realiza una clasificación (o regresión) en función de sus entradas así como de sus resultados. Por ejemplo, para cada nodo activo en la red TSN podría existir un árbol de decisión que seleccione, en función del estado del resto de nodos en la red (conocido a través de LLDP), la mejor configuración para un determinado flujo a través del uso de reglas compactadas y dividiendo así el conjunto de casos. Sin embargo, para el caso de redes neuronales ya no se hablaría por tanto de un aprendizaje por refuerzo basado en descenso del gradiente, sino de un algoritmo evolutivo trabajando sobre los pesos y conexiones entre las neuronas. Por tanto, DRL se encarga más de un modelo no supervisado a partir del coste de las acciones *online* sobre el entorno mientras que los algoritmos genéticos se aplican a problemas de optimización más amplios donde prima el resultado antes de llevar a cabo un despliegue de la red. Así, evaluando la misma topología en el caso de que un enlace falle, dichas reglas podrían conducir a las diferentes soluciones en que puedan rehacerse de ello para así ejecutarlas en cada nodo en el menor tiempo posible. Sin embargo, en este trabajo dejaremos a un lado estos modelos de clasificación para centrarnos únicamente en los algoritmos genéticos y su implementación para redes TSN con el fin de optimizar el rendimiento de éstas.

3.1.2. Solución propuesta para el modelo TAS

Como se ha podido observar en el anterior Apartado 3.1.1, existen múltiples esquemas para configurar e incluso optimizar la planificación de los tiempos de transmisión de los flujos en los distintos nodos que conforman una red TSN. Se ha visto por tanto que algunos son más complejos computacionalmente y elevan exponencialmente el tiempo de convergencia hacia una solución cuanto mayor sea el número de flujos al tratarse de un problema NP-duro, pero son capaces de garantizar una solución óptima en caso de que ésta exista. Otros, en cambio, son capaces de simplificar el problema, pero a cambio de una convergencia más arriesgada que puede estancarse en máximos locales. De este modo, cabe a destacar la importancia de minimizar la existencia de huecos para un uso más eficiente de los recursos de la red TSN, pero también el retardo extremo a extremo principalmente en aquellos con un límite de tiempo más ajustado, pues recordemos que existen también aplicaciones en las que, pese a tener un máximo retardo permitido, conviene minimizar además el tiempo de respuesta del sistema como podría ser el caso de *platooning* anteriormente expuesto. Con ello, se pretende implementar un planificador basado en algoritmos genéticos que optimice tanto el retardo *end-to-end* del conjunto de los flujos así como la eficiencia en el uso de los recursos a través de unos pesos para ambos parámetros, pues dependiendo de las aplicaciones con las que se pretenda trabajar puede ser más interesante tener en mayor consideración uno u otro; y además lo haga con valores de tiempo continuos para aprovechar lo máximo posible los recursos de la red TSN. Para ello se tratará la eficiencia en el uso de los enlaces del mismo modo en el que se presenta con la siguiente Ecuación 3.1.

$$Eficiencia(\,\%) = \left(1 - \frac{\sum_{i=1}^{Khuecos} hueco_i + \sum_{j=1}^{Mflujos} GB_i}{t_{fase}}\right) \cdot 100 \qquad (3.1)$$

Nótese que esta eficiencia queda normalizada respecto a los tiempos de fase para cada puerto, por lo que sus valores serán relativos al esquema presentado con los distintos periodos de todos los flujos que se pretendan planificar, así penalizando aquellos huecos en los que, en el peor de los casos, no quepa la inserción de una sola trama Ethernet. Del mismo modo, el retardo extremo a extremo también puede normalizarse respecto a su valor máximo tolerado, midiéndose así para todo el conjunto de flujos y evaluando los resultados del modelo sobre ambos parámetros. La intención de normalizar este valor en el caso de las latencias es que puedan optimizarse los resultados distinguiendo siempre los flujos en los que el retardo conseguido con el modelo queda más ajustado al tiempo máximo con el que una trama perteneciente a flujo puede llegar a su destino, pues de esta forma se puede lograr que una pequeña variación del primero suponga un mayor compensación en la evaluación respecto a otros en los que se goza de una mayor flexibilidad en cuanto a requisitos.

Con todo ello, se propone como solución a la planificación de flujos en TSN síncrono el desarrollo de un modelo TAS propio basado en el mismo esquema de asignación en híperperiodo y fases de tiempo continuo con el que poder evaluar ambos parámetros y con ello optimizarlos a través de un algoritmo genético a través de pesos, tratando a su vez reducir el coste computacional de éste.

3.2. Integración del planificador TSN con 5GS

Por otro lado, aparece también un importante problema relacionado con el hecho de emplear las unidades de RU, DU y CU además de los propios UPF dentro de una red de transporte 5G dado que se tratan de funciones virtualizadas que operan en capas superiores y por tanto introducen ciertas fluctuaciones temporales durante la computación de las tramas, dando lugar a ese *jitter*. Por ello, se requiere recurrir a soluciones como la propuesta en la sección IV.C 2 y 3 de O. Xinjian et al. [43], donde se trata un esquema de menor nivel a través de lo que se conoce como *Pseudo Time Synchronization Packet (PTSP)*, con el que se pretende ajustar estas variaciones temporales a tiempos fijos con el uso de tamaños de paquetes de tamaño dinámico en los que se inserta una serie de Bytes vacíos hasta el comienzo de la transmisión del siguiente, los cuales se han de descartar. Sin embargo, es posible que estos dispositivos no estén sincronizados, además de que esto provoca un desperdicio del ancho de banda del sistema, pudiendo llevar a cuellos de botella en redes con más tráfico; además de acabar afectando a los tiempos reservados para los diferentes flujos y por ende impidiendo el cumplimiento de sus requisitos de tiempo en cuanto a retardo. De igual forma, para el caso del RU, al tratarse de elementos implementados en las capas más bajas, dicha variación será realmente pequeña. En cambio, para el caso del DU, CU y UPF sí existe un importante problema ya que se tratan de NFVs que operan sobre capas superiores con el *split* ya visto en el Apartado 2.3, lo que implica que un sistema operativo (normalmente Linux) debe gestionar cierto número de procesos en paralelo y con ello los recursos hardware sobre los que operan éstas. De este modo, el tiempo de respuesta de estas funciones puede verse afectado por el método de gestión de las interrupciones de este sistema, con lo que puede existir una gran variabilidad que no se puede conocer a través de probabilidades. Estudios como el de S. Charalampos et al. [44] nos demuestran la posibilidad de hallar mediante DPDK configuraciones que nos permitan minimizar el tiempo de cómputo de estas funciones virtualizadas así como el *jitter* que provocan.

3.2.1. Solución propuesta para la integración de TSN síncrono con 5G

Es posible que el propio esquema de TSN síncrono también ofrezca una solución pues, tal y como se ha comentado anteriormente, aunque las bandas de guarda no son a priori necesarias entre flujos sensibles al tiempo, sí que pueden emplearse en la planificación del camino de dicho flujo en el nodo inmediatamente posterior a la NFV y que reciba estas tramas, con lo que con una previa cuantización absoluta de esta variación temporal (en función del flujo, principalmente del tamaño de la trama) a su llegada al nodo se puede iniciar una banda de guarda que preceda al siguiente flujo a planificar y en la que se incluya esa variabilidad de forma que no afecte a su transmisión, pues de lo contrario podría retardarlo. Con ello, en el siguiente nodo en el camino ya no sería necesaria esa banda de guarda posterior al intervalo de transmisión del flujo ya que, contando con que el inicio de ésta quedará fijado después del instante de llegada del límite de la banda de guarda, esta fluctuación quedaría completamente absorbida. Por tanto, esto solamente ocurriría en la planificación para aquellos flujos que llegaran a un nodo justo después de haber sido procesados por uno de estas NFVs, por lo que esto se produciría a lo largo de la propia red de transporte del *bridge* 5G y a su salida del UPF como puerto virtual, es decir, en su nodo vecino del siguiente salto, tal y como pueden verse marcados en rojo en la Figura 3.7. Nótese que de este modo, al haber sido absorbido el *jitter* en estos nodos, el resto no se verán afectados y podrán llevar a cabo la planificación sin necesidad de usar estas bandas de guarda.

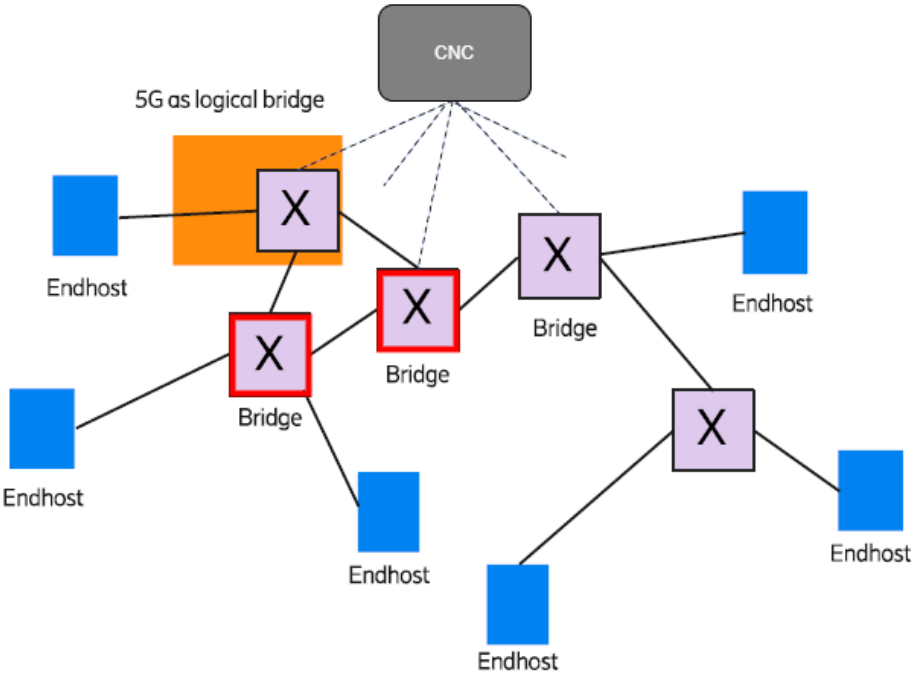


Figura 3.7: Nodos TSN afectados por indeterminismo de NFVs del *bridge* 5G [27]

Sin embargo, esto introduciría nuevos huecos que afectarían a este aprovechamiento de los recursos de la forma en que lo hacen en la Ecuación 3.1, pero a cambio solventarían este gran problema de sincronismo entre ambas tecnologías. Por ello, a dicha ecuación se le ha de añadir también la cantidad de tiempo destinada a salvar los intervalos de la siguiente transmisión mediante una guarda que se suma a las del sistema de TSN síncrono, todo

ello en proporción al tiempo de la fase. Por tanto, cuanto mayor sean estos tiempos en los que no se pueda transmitir otro tipo de tráfico, menor será la eficiencia de la planificación y con ello el *throughput*. Su valor total será la suma de la eficiencia para todas las fases planificadas de cada enlace. Es por este motivo por lo que es realmente importante tener en consideración tanto el retardo en la red de todos los flujos, en especial aquellos más críticos; como la utilización de los enlaces para permitir la transmisión de otras colas de prioridad dentro de la visión de la Industria 4.0 descrita en este trabajo. Sin embargo, dado que este problema se produce principalmente en la red de transporte del *bridge* 5G con el resto de NFVs, pese a que nos centraremos únicamente en su modelo de caja negra, es realmente importante considerar el uso de TSN síncrono también en la propia red de transporte para acotar aún más este *jitter*. No obstante, todo esto se tratará con mayor profundidad en este trabajo junto con la evaluación del retardo durante el desarrollo del sistema del Capítulo 5.

Capítulo 4

Algoritmos genéticos. Teoría e Implementación

4.1. Introducción a los Algoritmos Genéticos

Los algoritmos genéticos son un subtipo de los denominados algoritmos evolutivos, los cuales están muy extendidos para la resolución de problemas de optimización. Estos mecanismos están inspirados en la teoría de la evolución promulgada por el biólogo Charles Darwin en 1859 y que tratan de simular la adaptación de las especies al medio. Es a través del cruce entre individuos y la mutación de su genes lo que les garantiza a aquellos que mejor se adapten una mayor probabilidad de sobrevivir, ésto es, de ser los que prevalezcan frente al resto y transmitir a las posteriores generaciones su material genético. Su adaptabilidad se evalúa a través de una función de aptitud o *fitness*, la cual determina la calidad de las soluciones con un rendimiento sobre el entorno.

Estas soluciones pueden equipararse al concepto de cromosoma, el cual está compuesto de genes que adquieren una codificación determinada (genotipo), bien sea binaria, cualitativa o numérica. Estos genes vienen a representar de forma unívoca a cada una de las variables del problema (fenotipo), como podría ser una de las posibles rutas en la red TSN que sigue cada uno de los flujos. Con ello, dos o más individuos seleccionados pueden combinar sus genes tras la reproducción de modo que la siguiente generación herede una porción de cada uno de los cromosomas de la progenie. Nótese que se tendrán tantos descendientes como ascendentes dado que la porción de un cromosoma irá a parar a uno de ellos mientras la otra la adquirirá el otro. Además, es posible que en esta nueva generación a lo largo de su vida muten genes con cierta probabilidad, es decir, se modifiquen su valores o incluso se intercambien entre ellos. De esta forma, si se vuelve a evaluar a la nueva generación nos encontraremos con que existen individuos que, gracias al material genético heredado, han logrado adaptarse mejor al medio dado que su *fitness* ha aumentado; aunque también puede ocurrir lo contrario, que pese a que el nivel de adaptación de sus progenitores era bueno, su combinación ha hecho que su valor empeore. Los individuos de la nueva generación reemplazará a la anterior, aunque es posible que alguno perdure como el mejor de su generación, esto es lo que se conoce como elitismo y permite mantener el valor de *fitness* en el caso de que todos empeoraran. Con todo ello, si se repite este ciclo de selección, reproducción, mutación, evaluación y reemplazo durante un determinado número de generaciones o hasta lograr un *fitness* objetivo, el *fitness* se verá incrementado a medida que surgen individuos que mejoren la solución, dando así lugar a la optimización del problema. Un esquema de este proceso puede verse con la Figura 4.1.

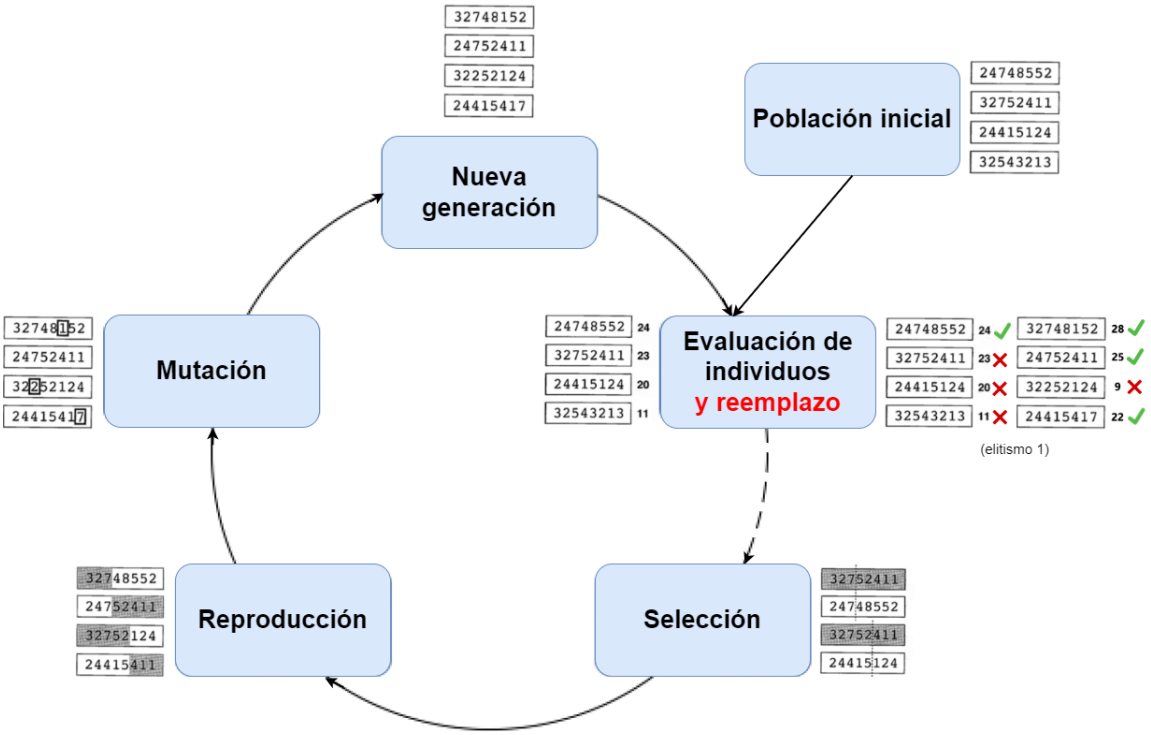


Figura 4.1: Ciclo de los algoritmos genéticos

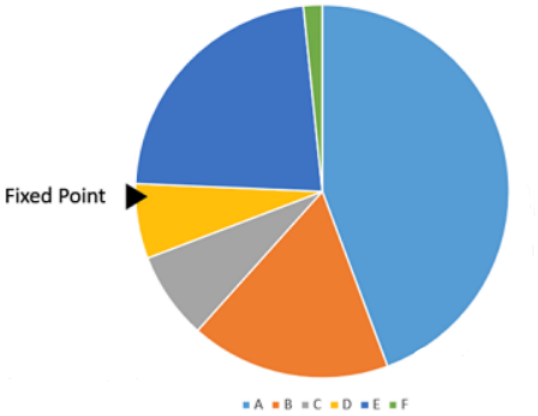
Sin embargo, la convergencia hacia la solución óptima depende principalmente de la selección de cuáles son los individuos que van a emparejarse para reproducirse, así como de otros factores como el número de genes que aporta cada uno de los progenitores o la probabilidad de que cada uno de éstos mute. A continuación se presentan estos mecanismos.

4.1.1. Mecanismos de selección de individuos

Como se ha dicho antes, cuanto mayor sea el *fitness* del individuo, mayor es la probabilidad de que sea seleccionado en la reproducción así elevando lo que se conoce como «presión selectiva», pero no siempre es así ya que individuos con menor adaptación pueden emparejarse con estos y originar una descendencia que mejore la solución. Entre los mecanismos de selección más utilizados se encuentran:

- **Selección aleatoria**. Los padres se eligen y emparejan aleatoriamente de entre el conjunto de la población. Este mecanismo permite mantener una presión selectiva baja al recombinar los peores casos con el resto de resultados, lo cual se traduce como oportunidad de una porción de su cromosoma que podría ser adecuada, aunque esto también podría impedir la reproducción de los mejores.

- **Selección por ruleta**. Se asigna una probabilidad de selección proporcional al valor del *fitness* de la solución sobre el resto de valores en una ruleta, la cual se hace girar para que caiga en uno de ellos y éste pueda ser seleccionado como padre de la nueva generación. Ya no se trata por tanto de un proceso puramente aleatorio, sino que existe mayor probabilidad de seleccionar los mejores resultados en la reproducción, aunque favorece más a los mejores sobre los demás y eleva así la presión selectiva.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Figura 4.2: Selección por ruleta

- **Selección por orden lineal o ranking**. Se trata de un método muy similar al anterior, donde también se hace girar una ruleta, solo que ahora la población se ordena en función de su valor *fitness* y cada individuo adquiere una probabilidad de selección que depende de su posición en la lista. Por ejemplo, para una población de seis individuos como en el caso mostrado en la Figura 4.2, el de mayor *fitness* sustituirá su valor por 6, el segundo por 5 y así hasta llegar al peor resultado, el cual obtendrá el valor de 1. De este modo, la probabilidad se recalcula sobre la suma de este nuevo valor *fitness*, con lo que soluciones con muy mal resultado pueden ser seleccionadas. De este modo se rebaja la presión selectiva para restar dominancia de los individuos mejor adaptados frente a los peores.

- **Selección por torneo**. Se escoge al individuo con mayor *fitness* de entre un subconjunto más pequeño escogido aleatoriamente de entre la población, como podrían ser los tres de la Figura 4.3. Se produce un efecto similar al mecanismo anterior, aunque el aumento de la presión selectiva aumenta con el número de individuos que participan en el torneo.

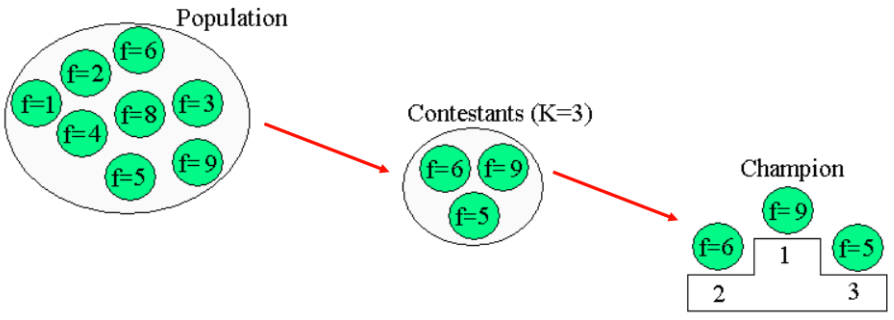


Figura 4.3: Selección por torneo

- **Selección jerárquica**. La presión selectiva se va aumentando a medida que aumenta el número de generaciones, variando así el mecanismo de selección entre los anteriores en determinados instantes del proceso.

4.1.2. Mecanismos de cruce entre individuos

Una vez seleccionados los individuos a emparejarse, el proceso pasa a observar la reproducción entre ellos. Esto se logra a través de la recombinación entre sus cromosomas. Es posible encontrar diversos tipos de cruce como los que se presentan a continuación:

- **Cruzamiento en un punto**. El cromosoma se divide en dos únicas porciones de genes en un punto, bien sea aleatorio o en función de los valores *fitness* de los progenitores. De esta forma, un descendente hereda una parte del cromosoma de un progenitor y la restante del otro hasta completar un mismo número de genes.

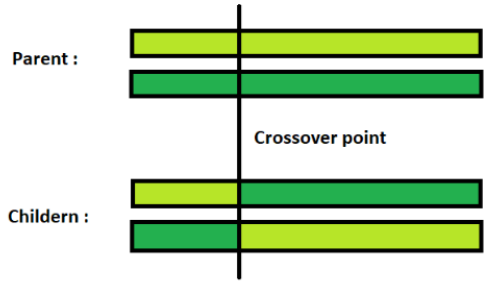


Figura 4.4: Cruzamiento en un único punto

- **Cruzamiento en n puntos**. El cromosoma se divide en $n+1$ porciones. Se asigna del mismo modo, pero debe haber al menos un gen por porción.

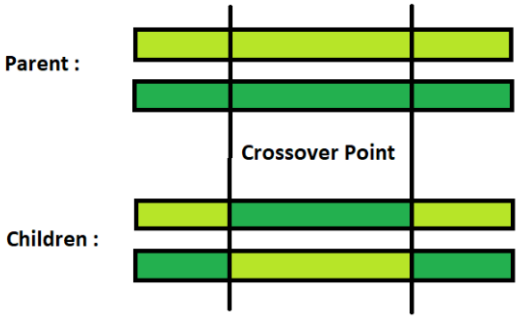


Figura 4.5: Cruzamiento en n=2 puntos

- **Cruzamiento uniforme**. El cromosoma se elige de forma aleatoria con cada uno de los genes entre ambos progenitores.

Figura 4.6: Cruzamiento uniforme

- **Cruzamiento aritmético**. Se realizan operaciones aritméticas entre pares de genes de las porciones de ambos cromosomas, como puede ser la suma, resta, valor medio, etc. Esto solo es válido cuando el espacio de soluciones admite dichos valores ya que, de lo contrario, una variable estaría adquiriendo un valor impropio del espacio de soluciones del que forma parte.
- **Cruzamiento OX**. De forma análoga al cruzamiento en n puntos, alguna de las porciones pasa a formar parte directamente del cromosoma de la nueva generación, mientras las otras porciones se establecen a través de un orden.

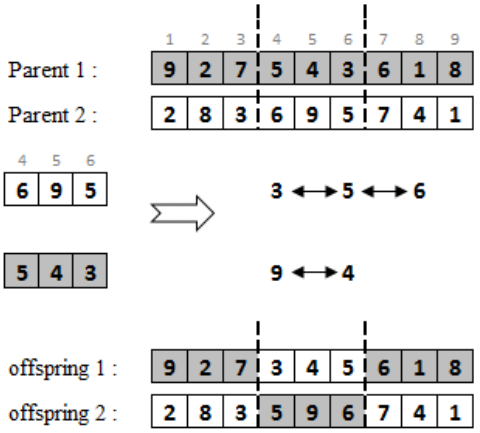


Figura 4.7: Cruzamiento OX [45]

Es a través de estos cruces entre individuos con los que el algoritmo irá obteniendo las posibles soluciones que mejoren los resultados al combinar los genes de cada uno de los padres. Sin embargo, de no seleccionarse una combinación adecuada entre los métodos de selección y cruce, puede llegar al punto en el que todos estos individuos acaben siendo muy similares entre sí por una alta presión selectiva sin llegar a mejorar el *fitness* así estancándose en un mismo resultado.

4.1.3. Mutación de los individuos

Una vez los individuos se han cruzado y formado la nueva generación, cabe la posibilidad de que cada uno de sus genes resultantes mute con una cierta probabilidad. Estos pueden mutar de diferentes maneras:

- **Mutación aleatoria**. Se produce cuando el gen mutable del individuo adopta un nuevo valor dentro del espacio de soluciones posible para tal gen. Dicho valor puede corresponderse con un valor entero o real dentro de una función de distribución de la probabilidad (e.g. gaussiana, uniforme etc.); pero también puede ser binario y únicamente mutar al valor opuesto en lo que se conoce como *bit-flipping*.

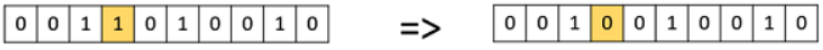


Figura 4.8: Mutación aleatoria

- **Mutación por intercambio**. Ocurre cuando el gen mutable intercambia su valor con otro gen dentro del mismo individuo, siempre que sea posible tal solución en la variable a la que representa.

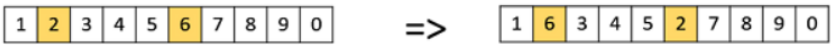


Figura 4.9: Mutación por intercambio

- **Mutación por reordenación**. Se realiza de forma análoga a la mutación por intercambio, solo que en este caso intervienen un mayor número de genes. Una variante de este tipo es la mutación por inversión, la cual se realiza con un subconjunto de genes y se intercambian sus valores entre las posiciones simétricas a un gen central.

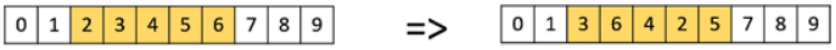


Figura 4.10: Mutación por reordenación

Estas mutaciones permiten explorar un mayor espacio de posibles soluciones cuando todos los individuos tienden a recombinarse en generaciones muy similares entre sí y donde con ello no hay una mejora de la población, lo que se conoce como saturación. Por tanto, bajas probabilidades de mutación por cada gen pueden conseguir esos pequeños cambios en el cromosoma que logren optimizar la solución, aunque esto también puede llevar a la convergencia hacia máximos locales al no poder diversificarse tanto. Por el contrario, una probabilidad más alta puede llegar a evitar esta situación, aunque también puede empeorar tal convergencia al no mutar únicamente los genes adecuados y por ello necesite un mayor número de generaciones hasta dar con la combinación más adecuada, sobre todo cuando existe un mayor volumen de variables representadas por los genes. Además, es muy importante tener en cuenta que, en el caso de intercambiar los valores entre los genes, es posible que el nuevo valor adquirido en alguno de ellos no se corresponda con el espacio de búsqueda disponible para tal variable, por lo que se hace necesario restringir estas acciones. De este modo, la vía más utilizada para estas mutaciones suele ser la aleatoria.

4.1.4. Mecanismos de reemplazamiento

Finalmente, una vez finalizado el proceso en el que la nueva generación de individuos ve modificado sus genes, es posible llevar a cabo diferentes estrategias para reemplazar a la generación anterior. Según éstas, se podrá decidir sobre qué trayectorias hacia la convergencia de determinadas soluciones son o no aptas. Entre ellas están:

- **Reemplazo de toda la generación**. En este caso se sustituyen todos los individuos de la anterior generación por la nueva formada después de todo el proceso de selección, cruce y mutación. De este modo ninguno de los individuos de la anterior generación sobrevivirá, aunque es posible establecer el elitismo de forma que la mejor solución de ésta evita su reemplazo con el peor resultado de la nueva generación, tal y como se ha visto en la Figura 4.1.

- **Reemplazo del peor**. También cabe la posibilidad de no reemplazar a toda la generación sino individuos concretos. En este caso se reemplaza únicamente el peor individuo de la anterior generación por su hijo en el caso de que éste mejore, elevando así la presión selectiva.

- **Reemplazo por torneo restringido**. Se reemplaza al padre más parecido a su hijo de un subconjunto de individuos elegido de forma aleatoria de entre toda la población, buscando así la mayor diversidad entre individuos que permita analizar un mayor número de trayectorias.

- **Reemplazo del peor entre semejantes**. Se remplaza la peor solución de un subconjunto de individuos de la anterior generación cuyos cromosomas se parecen más entre sí a los de su hijo para con ello descartar una trayectoria. Con ello se busca un compromiso entre mantener una menor presión selectiva y la diversidad.

- **Reemplazo de *crowding* determinístico**. El hijo de la nueva generación reemplaza al padre de la anterior generación más parecido para mantener la diversidad, aunque no sean resultado directo del proceso genético.

Sin embargo, el uso de estas técnicas en las que únicamente se reemplazan los padres más parecidos puede llevar a la necesidad de un mayor número de generaciones para converger en una solución optimizada.

4.1.5. Función *fitness*

La función *fitness* se trata del tipo de función objetivo aplicada a los algoritmos genéticos que mide la adaptabilidad de una solución al problema. Con esto, a través de la evaluación de cada uno de los individuos de cada generación, busca encontrar la combinación de genes que permiten optimizar el resultado. Para ello es necesario ejecutar un modelo que proporcione como salida un valor del parámetro a optimizar consecuente de tener como entrada los valores de las variables representadas en el cromosoma. Esta función es crítica ya que es a partir de ella con lo que se define la resolución del problema.

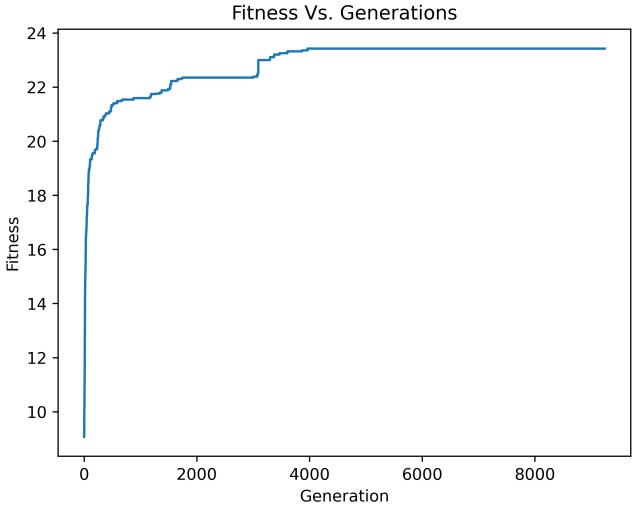


Figura 4.11: Evolución de la función *fitness* con el paso de generaciones

Un ejemplo de la evolución de esta función a lo largo de las generaciones puede verse con la Figura 4.11, donde se produce un crecimiento del valor obtenido de evaluar la población a medida que se la población evoluciona hacia soluciones mejor adaptadas al problema. Es a partir de cierto valor en el que dicha función deja de crecer para mantenerse en un mismo *fitness*, así saturándose con una población que tiende a ser idéntica entre sí. Por ello es importante limitar la ejecución de los algoritmos genéticos bien hasta conseguir un resultado satisfactorio (aunque no óptimo) o bien hasta un determinado número de generaciones en las que no se produce ningún cambio en su valor, por pequeño que pueda ser. Sin embargo, estos valores han de ser suficientes para permitir al algoritmo salir de situaciones como la que se presenta entre las 2.000 y las 3.000 generaciones, donde el valor *fitness* queda estancado hasta dar con una solución que mejora el resultado. Asimismo, en cualquiera de los casos ha de estar acotado por un número máximo de generaciones de modo que se limite el tiempo de cómputo del algoritmo.

Esta función *fitness* se encarga por tanto de optimizar el rendimiento sobre el entorno del problema para un único parámetro, aunque también puede trabajar con varios de ellos a través de pesos asignados que den mayor relevancia a aquellos más críticos. Con ello, esta función pasaría a tratarse de un sistema lineal con múltiples entradas para las variables del cromosoma a evaluar y múltiples salidas, cada una representando el valor resultante de uno de los parámetros de interés.

Además, como se ha ido comentando a lo largo de este Capítulo 4 dedicado a los algoritmos genéticos, una mala combinación de los métodos del proceso evolutivo puede desencadenar en la convergencia hacia máximos locales que pueden distar de la mejor solución posible. En la Figura 4.12 puede verse el caso en el que, con tan solo dos variables, se pueden producir máximos locales por debajo de la solución óptima para el problema. Claramente, este espacio de búsqueda será más complejo a medida que se vayan sumando nuevas variables, como en nuestro caso puede ser el número de flujos y, con ello, las posibles combinaciones entre rutas para minimizar tanto el retardo como los huecos.

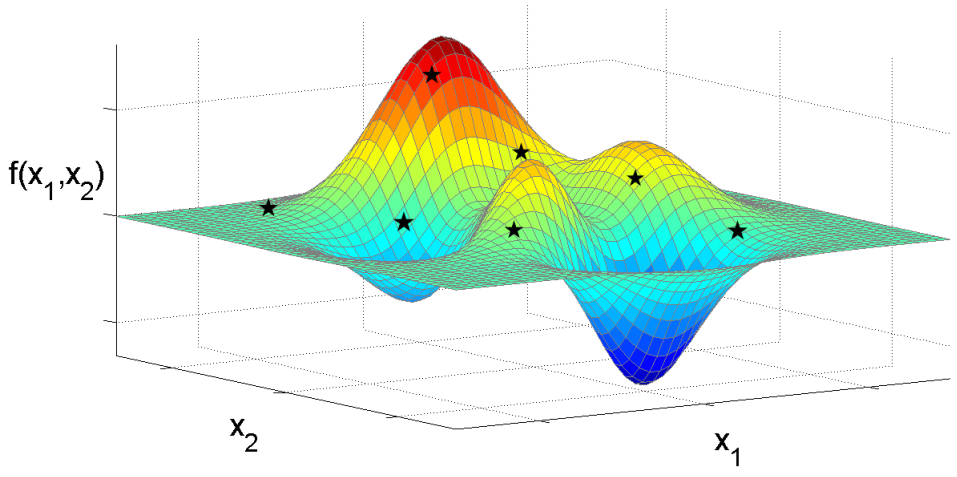


Figura 4.12: Convergencia hacia máximos en el espacio de búsqueda

Por tanto, este tipo de algoritmos son muy útiles para resolver problemas complejos en los que no puede definirse una función matemática habitual dada su aleatoriedad o discontinuidad y que permita optimizar los resultados. Además, presentan multitud de

variantes a través de los parámetros expuestos que permiten orientar el problema de una forma u otra en función de las necesidades. Los algoritmos genéticos permiten por tanto profundizar sobre el espacio de soluciones sin restricciones, aunque esto posee la peculiaridad también de que no se garantiza una solución óptima en un tiempo o número de generaciones limitado como podría ser por ejemplo la saturación en la función *fitness*. Con ello, cabe la posibilidad de que transcurran generaciones hasta tal punto en el que toda la población posea características muy similares entre sí llegando incluso a ser iguales y tan solo mutando algunos genes sin llegar a lograr mejores soluciones, lo que puede llevar también a máximos locales cuando la solución óptima dista de la convergencia del algoritmo sobre los genes cruciales. Conocidos problemas similares como el «viajante de comercio» emplean algoritmos genéticos para la optimización de la solución, donde dependiendo de la inicialización puede desencadenar en una solución subóptima distinta en cada ejecución cuando el problema es más complejo, por lo que se trata de optimizar. Es por todo ello por lo que se ha decidido en este proyecto llevar a cabo un nuevo *framework* basado en algoritmos genéticos que sea capaz de aprender una configuración más optimizada del conjunto de la red TSN sobre el transporte del *bridge* 5G así como de la red TSN completa de forma *offline* a través de un modelo que sea capaz de seleccionar el mejor camino y en base a estos tiempos dados unos requisitos QoS para los diferentes flujos existentes.

Será en la Sección 5 con el diseño del sistema donde se profundizará más en este trabajo, no sin antes comentar las herramientas software y hardware empleadas para la implementación del modelo junto con el algoritmo genético.

4.2. Herramientas para la implementación

A continuación se procede a realizar un breve análisis de las herramientas utilizadas en este proyecto, entre las que se encuentran el lenguaje de programación Python junto con el uso de la librería PyGAD y el *framework* de trabajo de Microsoft Visual Studio en lo que a software respecta así como el supercomputador de la Universidad de Granada en cuanto a hardware principal para su ejecución.

4.2.1. Python 3.9 en Visual Studio Code

Python se trata de un lenguaje de programación de código libre creado a finales de los años 80 por el informático holandés Guido Van Rossum como sucesor del lenguaje *ABC*. Python es un lenguaje interpretado, por lo que las instrucciones se traducen a código máquina y se llevan a cabo una a una conforme se van ejecutando secuencialmente. Esto permite poder implementar código sobre cualquier plataforma, lo cual implica indirectamente la portabilidad del mismo entre diferentes máquinas sin necesidad de reprogramarlo en función de sus características. Sin embargo, dado que no hay una compilación que se traduzca a la ejecución de código máquina, el tiempo de cómputo en las tareas realizadas aumenta respecto a los lenguajes compilados, como puede ser el caso de *C++*. Por otro lado, al ser un lenguaje de alto nivel muy sencillo, permite que el programador pueda desarrollar y comprender con mayor facilidad el código dada su gran proximidad con el lenguaje humano. Además, está orientado a objetos, por lo que permite crear sistemas más complejos y agiliza el desarrollo mediante la reutilización y ampliación de código gracias a la capacidad de abstracción que ofrece. Sin embargo, la mayor ventaja que Python ofrece es la posibilidad de acceder a un gran volumen de librerías que la comunidad ha

ido desarrollando con el fin de facilitar aún más el proceso de desarrollo, pues basta con importarla y hacer uso de cualquiera de sus funciones. Esto es común a la mayoría de los lenguajes de programación, pero aquí Python cuenta con un gran número de librerías muy optimizadas destinadas al análisis y minería de datos con *Big Data* e Inteligencia Artificial. Es por este motivo por el que el ámbito laboral está demandando cada vez más programadores especializados en estos campos a través de Python para el desarrollo de nuevas funciones, pues tampoco requiere de licencias. Según publica la revista *IEEE Spectrum* en su edición de 2021, Python es el lenguaje más extendido por su versatilidad. En la Figura 4.13 puede verse la clasificación presentada, donde Python se sitúa por delante de otros lenguajes muy dilatados como Java o *C++*, aunque es cierto que tiene sus límites ya que algunos están más enfocados a la programación de sistemas embebidos de tiempo real, otros a aplicaciones móviles, etc.; y ofrecen un mejor rendimiento en un campo concreto. En cambio, en el caso en el que nos centramos en este proyecto, el de la Inteligencia Artificial, Python es el más adecuado en su versión actual 3.9. En concreto, dado que se trabajará principalmente con algoritmos genéticos, se ha decidido hacer uso de la librería *PyGAD*, la cual se procede a comentar a continuación. Para ello, se hará también uso del editor de código fuente abierto *Visual Studio Code* de la compañía Microsoft, la cual ofrece un entorno de Python bastante completo en su interpretación. Además, cuenta con extensiones para facilitar la tarea, principalmente con aspectos como la depuración y la detección de potenciales errores.

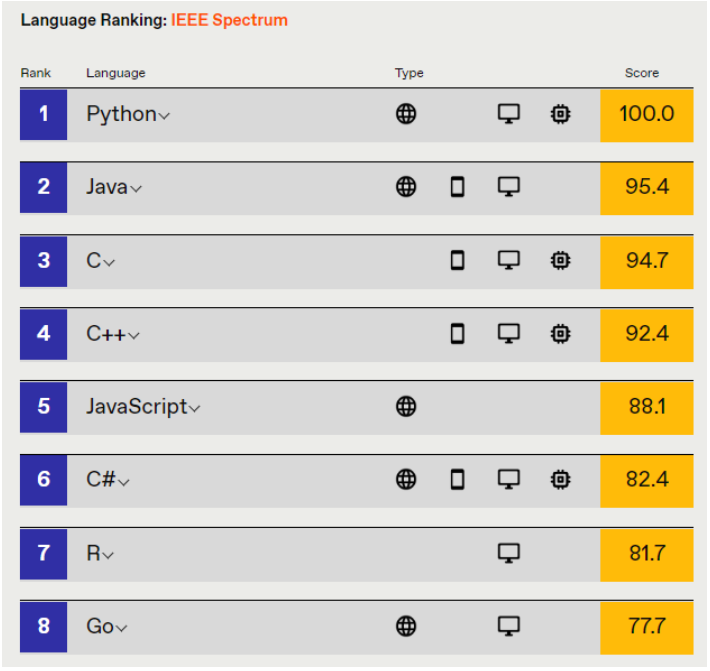


Figura 4.13: Ranking de lenguajes de programación

4.2.2. Librería PyGAD. Algoritmos genéticos en Python

PyGAD[46] es una librería de código abierto creada en Python por Ahmed Fawzy Gad para el desarrollo de algoritmos genéticos y *Machine Learning*, con lo que permite crear un *framework* de aprendizaje automático. De hecho, con él también se pueden crear redes neuronales basadas en las plataformas de Keras y PyTorch con el fin de poder inferenciar

una salida en base a las entradas que proporciona el algoritmo genético como mejores individuos o cromosomas. Sin embargo, como ya se ha explicado con anterioridad, esto queda fuera del foco de este proyecto puesto que nos centramos exclusivamente en el mecanismo del algoritmo evolutivo que seleccione la mejor solución para una situación dada. Por otro lado, permite la selección de diferentes configuraciones del algoritmo genético a través de los parámetros vistos en el Apartado 4.1, así pudiendo modificar el tipo de selección, el tipo de cruce, la población inicial, la probabilidad de mutación, etc. Además, es posible configurar funciones propias que se ejecuten en cada proceso visto en la Figura 4.1, entre ellas la función *fitness* que se encargará de evaluar tanto el retardo como el uso de los recursos en la red TSN. De este modo, podemos centrar la mayor parte del esfuerzo en el algoritmo de planificación. En la Figura 4.14 puede verse el organigrama de PyGAD.

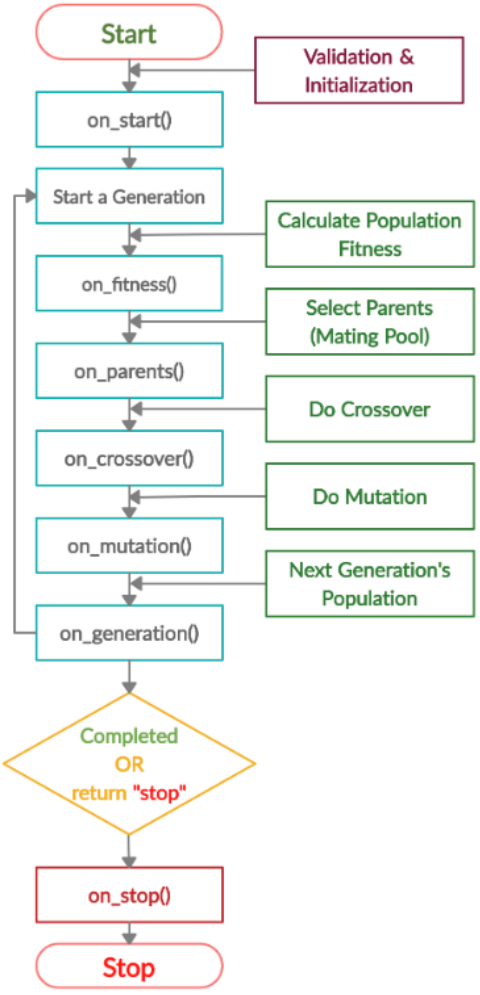


Figura 4.14: Organigrama de algoritmos genéticos con PyGAD

Con ello, se tendrá una evaluación de todos los nuevos individuos en cada generación hasta llegar al criterio de parada. Entre los parámetros más interesantes que en la actual versión 2.17 encontramos para el trabajo que nos atañe están:

- *num_generations*: Número máximo de iteraciones del proceso evolutivo. El criterio de parada, de no establecerse o no cumplirse algún otro, hará que el programa termine ofreciendo la mejor solución encontrada, es decir, aquella con mayor valor *fitness*.

- *stop_criteria*: Criterio de parada. Puede ser bien por sobrepasar un umbral de valor *fitness* o bien por saturarse dicho valor durante cierto número de generaciones. De no establecerse ninguno, el proceso finaliza cuando llega al número de generaciones marcado por el anterior parámetro. Es importante adaptarlo al ritmo de evolución del problema.

- *num_parents_mating*: Número de soluciones que se emparejan para dar lugar a otras dos nuevas. Normalmente se mantiene en dos, aunque también es posible elevarlo en el caso de aumentar también el número de porciones en el cruce.

- *initial_population*: Vector de población de soluciones inicial. El algoritmo genético suele mantener el número de soluciones ya que el cruce se realiza en parejas. Con ello, se pueden introducir valores aleatorios en primera instancia de modo que no se limite en exceso el espacio de búsqueda.

- *keep_parents*: En caso de establecerse un valor, mantiene la mejor solución de la generación anterior sobre los descendientes, lo que conlleva eliminar de éstos últimos aquellos que ofrezcan un peor resultado. Es adecuado emplear un único padre como elitismo.

- *num_genes*: Número de genes por cromosoma. Su valor ha de ser igual al número de variables que intervienen durante la evaluación.

- *gene_space*: Especifica para cada variable el espacio de soluciones que puede adquirir, por lo que tanto en los emparejamientos como en las mutaciones solo puede modificar una variable siempre y cuando esté dentro de dicho espacio. Es posible por tanto acotar el espacio de búsqueda a través de este parámetro.

- *gene_type*: Especifica para cada variable el tipo de valor que puede adquirir. Por ejemplo, puede tratarse de un número entero, un real, vector, etc. La función *fitness* es la encargada de interpretar dicho valor.

- *parent_selection_type*: Tipo de selección entre soluciones. PyGAD permite escoger entre métodos como el de ruleta, ranking, torneos o aleatorio. Adicionalmente, se puede implementar un algoritmo personalizado que aplique otros métodos entre los ya estudiados.

- *crossover_type*: Tipo de emparejamiento llevado a cabo. Entre ellos, PyGAD permite escoger entre los métodos de único punto, dos puntos, uniforme y aleatorio. Adicionalmente, se puede usar también una función propia, con lo que cabe la posibilidad de emplear otros métodos diferentes como los ya vistos anteriormente.

- *mutation_type*: Tipo de mutación de las soluciones. Con PyGAD se puede escoger entre aleatorio, intercambio o mezcla. Adicionalmente, se puede implementar un algoritmo personalizado, aunque lo más común es usar mutación aleatoria con una probabilidad determinada para recorrer más el espacio.

- *mutation_probability*: Probabilidad de mutación de cada gen en una misma solución.

Finalmente, existen otros muchos parámetros con los que se pueden lograr configuraciones más enfocadas a las características concretas de cada problema. En el Capítulo 5 se verá con más detalle el uso de estos parámetros así como su implicación en los resultados de este proyecto.

4.2.3. Supercomputador de la Universidad de Granada

Una vez conocido ya el software empleado para el desarrollo del modelo evaluado por el algoritmo genético, dado el alto coste computacional que conlleva su ejecución, se ha decidido recurrir al servicio de supercomputación que ofrece la Universidad de Granada para investigadores y empresas asociadas, el cual cuenta con una infraestructura amplia destinada al cálculo intensivo de procedimientos y simulaciones.

Características técnicas

La ampliación más reciente para esta infraestructura de supercomputación es de febrero de 2022 con la unidad denominada «Albaicín». Ésta se suma a la ya existente desde 2007 y que llegó a estar en la lista *TOP500* como uno de los más potentes a nivel mundial, «UGRGrid»; junto con su extensión de 2013, «Alhambra». En la siguiente Tabla 4.1 se muestra una comparación entre ellos.

Unidad	Núcleos	Procesador	Conexionado	Cálculo	Memoria RAM	Almacenamiento
UGRGrid	1264	Opteron Dual Core 2C 2.2 GHz	Gigabit Ethernet (1 Gbps)	4.2 Tflops	3 TB	24 TB
Alhambra	1808	Intel Xeon E5-2680 2.7 GHz	Infiniband QDR (32 Gbps)	36.75 Tflops	4.28 TB	72 TB
Albaicín	9520	Intel Xeon Gold 6258R 2.7 GHz	Infiniband HDR (200 Gbps)	822 Tflops	35 TB	184 TB

Tabla 4.1: Unidades del supercomputador de la Universidad de Granada

El equipo Albaicín ofrece por tanto un rendimiento unas veinte veces superior a la unidad anterior Alhambra (y doscientas veces respecto a UGRGrid), lo que ahora sitúa a este supercomputador como la mayor infraestructura de cómputo de Andalucía y entre los diez más potentes del país. Es por este motivo por el que es capaz de prestar servicio multidisciplinar a más de 500 investigadores, quienes trabajan en líneas de nanopartículas, astronomía, estructuras biomoleculares o Inteligencia Artificial. En concreto, pone a disposición de un total de 170 nodos que pueden ser utilizados por los investigadores de forma paralela para los procesos, lo que permite reducir considerablemente el tiempo de cómputo y con ello progresar en el estudio de procesos aún más complejos y con mayor precisión. Además, se dispone de recursos ampliados en cuanto a capacidad de almacenamiento de los resultados de estos procesos y simulaciones así como de un canal seguro para la transferencia de ficheros mediante *SSH File Transfer Protocol (SFTP).*

Acceso SFTP y planificación de recursos con SLURM

De este modo, los investigadores pueden acceder a través de cualquier cliente SFTP a su directorio para organizar los ficheros necesarios (e.g. código, bases de datos, muestras de voz, imágenes, etc.), designar los recursos de cómputo necesarios para llevar a cabo la tarea, ejecutar el proceso y finalmente disponer de los resultados. Para ello, la Universidad de Granada dispone de una plataforma web de código abierto en la que poder realizar el envío de los trabajos a ejecutar a través del sistema de gestión de tareas y recursos *Simple*

Linux Utility for Resources Management (SLURM), centrándose así únicamente en la administración de estos recursos a través de colas de trabajos independientes. Está claro que, si un trabajo puede ejecutarse de forma paralela en varios nodos se logrará reducir el tiempo de cómputo, pero también puede llegar a limitar el acceso a otros procesos de igual prioridad hasta que el anterior haya finalizado, por lo que debe administrar correctamente tales recursos. Por tanto, a un usuario se le puede permitir reservar los recursos a través de parámetros en función de sus necesidades aunque, por el contrario, es el sistema el que en todo momento decide cuáles son los elementos utilizados. Todo esto podría verse como la comanda en un restaurante, donde el camarero toma nota de lo que el cliente le solicita. Una vez tiene anotado el pedido, marcha a cocina con la nota para que su plato aguarde en cola hasta que finalmente es preparado y servido sin que el cliente sepa exactamente si su plato ha sido cocinado en una sartén más grande o si en cambio han utilizado otras más pequeñas, si lo han preparado de forma individual o lo han hecho atendiendo también a las demandas de otros clientes; pues lo único que le interesa es que le llegue cuanto antes y no tanto el cómo lo hacen. Pues bien, el sistema SLURM hace algo parecido, salvo que esta vez la comanda se hace a través de ficheros ejecutables *Bash*, en los cuales se definen los parámetros para dicha reserva y que SLURM interpreta a través de comandos *SBATCH* incluidos en este fichero. Entre ellos están el número de nodos, la memoria RAM empleada, el tiempo dedicado, etc. A continuación puede verse con las siguientes líneas un caso real que podría utilizarse durante este proyecto.

Listing 4.1: Fichero Bash con comandos SBATCH para la reserva de recursos en SLURM

```
#!/bin/bash
#SBATCH --job-name = genetic_algorithm_PRM
#SBATCH --mail-type = END,FAIL
#SBATCH --mail-user = pablorodrimar@correo.ugr.es
#SBATCH --nodes = 1
#SBATCH --partition albaicin
#SBATCH --mem = 2gb
#SBATCH --time = 24:00:00
#SBATCH --output = /path-to/work/folder/logs.out

hostname; date

source /path-to/python_venv/bin/activate

python3 /path-to/python_script.py >> /path-to/results.txt

date
```

Como se puede observar, en este caso se está reservando un único nodo, el cual cuenta con un total de 56 núcleos, sobre la partición del supercomputador «Albaicín» y con 2 GB de memoria RAM. Además, se está limitando el proceso a un máximo de 24 horas, pues de lo contrario adquiriría por defecto el valor máximo permitido por tal partición y no sería adecuado para la administración de tareas en SLURM. Tras ello, el propio fichero *Bash* se encarga de lanzar el código implementado en Python para guardar los resultados en un un documento dentro del mismo directorio habilitado para el usuario. La mayor ventaja de esto es que es posible paralelizar los procesos en diferentes tareas para reducir el tiempo de cómputo y con ello obtener sus resultados en la mayor brevedad posible. Por último, cabe a destacar la importancia de controlar la duración de los procesos, pues en Inteligen-

cia Artificial es necesario estudiar la viabilidad de los algoritmos empleados además del resultado que puedan ofrecer, pues de esta forma pueden tratar de optimizarse y controlar el tiempo de espera hasta que un proceso termina para analizar sus resultados. Para ello, también es posible configurar la tarea de modo que nos avise por correo electrónico cuando el proceso haya finalizado o si se ha llegado a producir un error.

Adicionalmente, SLURM cuenta con otros comandos de usuario para controlar dichas tareas, comprobar el estado de un proceso, lanzar nuevos trabajos, cancelarlos, etc.; según el administrador permita. En la Figura 4.15 puede apreciarse un diagrama con el sistema de componentes generalizado de SLURM, donde cada uno de los nodos está controlado por un programa residente, que a su vez está gestionado de forma jerárquica por el controlador principal así comprobando en la base de datos aspectos como los permisos de usuario, los procesos activos, etc. Este controlador es el encargado de comprobar los requisitos de cada tarea y el estado de los recursos con el fin de asignárselos en función de éstos así como de llevar a cabo los comandos que el usuario le indica.

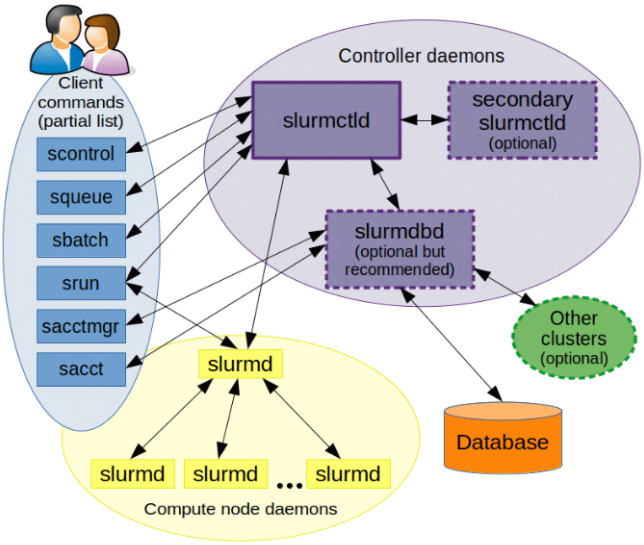


Figura 4.15: Sistema de componentes en SLURM

Por ejemplo, con el comando *squeue* de la Figura 4.16 se puede comprobar el estado de los procesos de un usuario, como podría ser el caso del lanzado con el anterior fichero. De esta forma, podemos conocer la duración hasta ahora desde su inicio así como el identificador del proceso o el nodo en el que se está ejecutando. En el caso de que todos los nodos estuvieran siendo utilizados por los procesos de otros usuarios, la tarea permanecería en cola hasta que los recursos necesarios quedaran libres y el algoritmo de optimización de SLURM así lo decidiera en función de los requerimientos (tiempo, número de nodos, memoria RAM, permisos del usuario, etc.).

```
JOBID     PARTITION        NAME      USER     STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
26221      albaicin genetic_algo pablorodri   RUNNING       0:23 1-00:00:00      1 hcn150
```

Figura 4.16: Ejecución del comando *squeue* en SLURM

Una vez definidas las herramientas que se emplean en este proyecto, pasamos al desarrollo del sistema de un planificador TSN optimizado a través de algoritmos genéticos.

Capítulo 5

Diseño del sistema

Conociendo ya con mayor profundidad el estado del arte acerca de los sistemas de TSN y 5G así como las herramientas necesarias para su integración en un escenario más orientado a la Industria 4.0, se puede comenzar ya a plantear una solución en base a las características de ambas tecnologías con el fin de permitir el transporte de los *slices* de 5G en redes deterministas como es el caso de TSN a fin de conseguir que las comunicaciones más críticas puedan cumplir con sus restricciones. Para ello, hemos destacado los siguientes aspectos:

I. Existe una gran variedad de algoritmos de planificación en redes TSN, principalmente heurísticos y metaheurísticos, aunque aquellos que pretenden realizar compresión para el mayor aprovechamiento de los recursos al eliminar estos huecos hace que la complejidad así como el tiempo de cómputo aumenten exponencialmente. Los algoritmos genéticos pueden ser una gran herramienta para solventar el problema en el que la planificación de los enlaces va intrínsecamente ligada a la ruta de cada uno de los flujos, donde no existe una función clara que lo defina. También influyen notablemente en este problema las restricciones de cada uno de los flujos, como pueden ser el periodo de transmisión, el retardo máximo permisible, el tamaño de las tramas, etc. Sin embargo, esto trae consigo un alto coste computacional, por lo que se ha decidido recurrir al diseño de un nuevo modelo más novel que simplifique esta planificación con el fin de poder comprobar resultados. Además, la configuración de los algoritmos genéticos no es trivial dado que el uso de las mecánicas de selección y reproducción u otras depende principalmente del problema.

II. El acceso radio 5G es el elemento en la red que, naturalmente, mayor retardo y *jitter* introduce, algo que rompe por completo el concepto de redes deterministas tal y como lo hemos estudiado en este trabajo. Existen métodos basados en modelos matemáticos e Inteligencia Artificial para la planificación de los recursos radio de modo que se logre minimizar ambos fenómenos ocasionados por la variación del canal. Sin embargo, nos encontramos que el problema radica ahora en las fluctuaciones ocasionadas por la computación de las tramas en NFVs. Con ello, este trabajo se enfoca en un ámbito más futurible en el que se puede lograr reducir dichas fluctuaciones sobre el orden del microsegundo de modo que tan solo sea necesario asignar una banda de guarda para resolver este problema, pues de lo contrario TSN síncrono no puede operar con 5G y se encontrará con un *bridge* lógico 5G empleando *hold-forward* para esperar los tiempos designados a cada cola, algo muy inestable en este tipo de redes.

Parece que trabajos más enfocados en la arquitectura de computadores basados en DPDK logran resolver esta problemática para servicios virtualizados.

III. Aunque se ha visto la posibilidad de integrar TSN síncrono dentro de la red de transporte de 5G, en muchas publicaciones se simplifica el *bridge* 5G como una «caja negra», pues los propios nodos que conforman su red de transporte introducirán un mayor retardo y elevarán el *jitter*. Por el contrario, dicha red de transporte basada en redes deterministas contribuiría a minimizarlos, pudiendo llegar directamente a formar parte de una red TSN completa gracias a la virtualización de la red industrial. Es principalmente por este motivo por el que resulta muy interesante tal integración. Sin embargo, se ha optado por esta reducción ya que actualmente no existe un módulo en la red de 5G que permita la integración de esta tecnología con TSN en su red de transporte, pues tan solo se pueden encontrar contadas publicaciones al respecto y en las que únicamente se analizan y proponen tales mecanismos como arquitectura novel para la interconectividad entre ambos núcleos.

IV. No existe un mapeo claro entre *slices* de 5G y clases de TSN. Gran parte de la bibliografía trata el tráfico más crítico como una única cola frente al resto del tráfico más o menos prioritario y el *best-effort*, independientemente de los periodos. De este modo, conociendo que pueden existir diferentes *slices* URLLC en función de las necesidades, pueden identificarse mediante un único PCP siempre y cuando pertenezcan a comunicaciones críticas. El resto de clases pueden ser asignadas en función de su prioridad, así concediéndole más tiempo de transmisión a aquellas que así lo requieran y dejando la última parte al tráfico *best-effort*. Aquí se ha destacar por tanto el uso de las bandas de guarda previas al tráfico sensible al tiempo de modo que no se colisione con sus tramas y su evaluación en la función *fitness* para penalizar la aparición de huecos en los que no se pueda transmitir nada.

V. Tampoco se ha encontrado ninguna publicación acerca del planificador TSN síncrono que trate de forma conjunta la optimización en cuanto a retardos y el aprovechamiento de los recursos de la red con el uso de las bandas de guarda en tiempo continuo como solución a las fluctuaciones de las NFV que participan en la red de 5G. Tan solo se han encontrado modelos de tiempo discretizado en *slots* no consecuentes con dichas fluctuaciones. En cambio, todo ello es muy importante en escenarios en los que se pretende minimizar el tiempo de respuesta para lograr una mayor eficiencia, como es el caso de *platooning* o la conducción autónoma de vehículos.

VI. Respecto a la capacidad de respuesta frente a posibles fallos, nos hemos encontrado con mecanismos que TSN trae ya consigo como es el caso de FRER, aunque también existen propuestas alternativas que podrían converger rápidamente sin necesidad de inundar la red. Sin embargo, mediante una planificación *offline* se puede llegar a lograr una respuesta aún más rápida conforme se detecten los fallos en la red, aunque también conlleva un importante coste computacional prever tales situaciones cuanto más compleja sea la red y mayor sea el volumen de flujos a planificar.

VII. Existen simuladores como OMNeT++[47] capaces de emular estos conmutadores TSN con buena precisión dado que poseen un mayor enfoque sobre la Industria 4.0 con el *framework* de INET. Sin embargo, este tipo de simuladores no son adecuados para optimizar la solución del problema mediante algoritmos genéticos dado el mayor tiempo de cómputo requerido para la obtención de estadísticas así como la dificultad de lanzarlo en el supercomputador ya que para ello es necesaria una interfaz gráfica.

En definitiva, se han identificado una serie de características que influyen directa o indirectamente en la adopción de soluciones. De este modo, se procede al desarrollo en base a estos requerimientos de un modelo de planificador TSN síncrono en Python que sea ejecutado y evaluado por un algoritmo genético a fin de optimizar los recursos de la red y minimizar el retardo. A lo largo de este Capítulo 5 se irá definiendo con mayor profundidad la metodología empleada así como su utilidad en el sistema.

5.1. Desarrollo del modelo TAS

5.1.1. Caracterización del tráfico

Según publica 5G-ACIA en [5], existen múltiples tipos de tráfico en la Industria 4.0 que, según su naturaleza, pueden poseer unas características u otras. Estos tipos de la Tabla 5.1 reflejan las necesidades de las aplicaciones que intervienen en esta nueva industria y que se han visto en el Apartado 2.1.

Tipo	Periodo	Long. (Bytes)	Critic.	PCP	Tol. jitt.	Tol. pérdidas	TAS/ATS
Isócrono	100 μs - 2 ms	30 - 100 *(f)*	Alta	6	0	No	TAS
Síncrono	500 μs - 1 ms	50 - 1000 *(f)*	Alta	5	$\leq \tau$	No	TAS
Asíncrono	2 ms - 20 ms*	50 - 1000 *(f)*	Alta	5	$\leq \tau$	1 - 4 tramas	ATS
Eventos: Control	10 ms - 50 ms*	100 - 200 *(v)*	Alta	4	-	Sí	ATS
Eventos: Alarmas & comandos del operador	2 s*	100 - 1500 *(v)*	Media	3	-	Sí	**
Control de red	50 ms - 1 s	50 - 500 *(v)*	Alta	7	Sí	Sí	-
Configuración & diagnosis	-	500 - 1500 *(v)*	Media	2	-	Sí	**
Vídeo	tasa imagen	1000 - 1500 *(v)*	Baja	1	-	Sí	**
Audio/Voz	tasa muestras	1000 - 1500 *(v)*	Baja	1	-	Sí	**
Best-effort	-	30 - 1500 *(v)*	Baja	0	-	Sí	**

* Periodos variables acotados, ** ATS opcional, f ≡ tamaño fijo, v ≡ tamaño variable

Tabla 5.1: Tipos de tráfico industrial de automatización en la red TSN [5]

Como se puede observar, existen dos casos en los que es necesario hacer uso del TAS: isócrono y síncrono. El primer caso se centra en un uso más restrictivo frente a *jitter* y retardo, pues la latencia está fijada al valor del periodo y su variación ha de ser nula. Con ello, este tipo de tráfico suele reservarse para comunicaciones con una mayor prioridad sobre el resto, por lo que suele relacionarse más con el uso de tramas destinadas a aplicaciones en las que se requiera la sincronización entre equipos a través de IEEE 802.1QAS. De este modo, su periodo puede llegar a reducirse hasta los 100 microsegundos. En cambio, para el caso del tráfico síncrono sí que se tolera un retardo extremo a extremo superior al valor del periodo y una cierta variabilidad acotada. Su periodo en cambio ha de ser superior a los 500 microsegundos e inferior a 1 milisegundo. A ello se le añade otro tipo de comunicaciones periódicas de alta criticidad como son los mensajes de control de la red, aunque éstos pueden llegar a periodos de hasta 1 segundo y son tolerantes al *jitter*. Normalmente se relacionan más con la gestión de los nodos que conforman la red TSN

para una rápida convergencia en caso de fallo, razón por la que adquieren la mayor de las prioridades a través de su PCP. Tanto en el caso isócrono como en el síncrono el tamaño de las tramas es fijo, por lo que los tiempos asignados a cada uno de estos flujos no varían. Sin embargo, en el caso destinado a mensajes de control su tamaño es variable, por lo que el intervalo de tiempo asignado debe ser tal que permita en todo momento la transmisión del tamaño máximo. Esto conlleva a que cuando se transmitan tramas de menor tamaño a dicho límite no se estén aprovechando del todo los tiempos reservados, por lo que aquí es más común encontrarse con un esquema de TSN asíncrono.

Por otro lado, aparecen un caso asíncrono, en el que se transmiten tramas de forma esporádica pero con un periodo variable acotado y con una mayor tolerancia en cuanto a pérdidas; y otro destinado a eventos de control entre dispositivos en los que existe mayor flexibilidad. Sin embargo, ambos pueden regirse por un esquema ATS en el que exista *preemption* (IEEE 802.1Qbu/br), por lo que no son objeto de estudio de este trabajo.

Existen además otras categorías caracterizadas por una menor prioridad y que podrían incluirse en otras colas de menor PCP, con lo que ya no es necesario que cumplan con unos tiempos, pero sí es posible reservarles unos determinados intervalos a cada una de estas colas de modo que las de criticidad media esperen menos tiempo en cola. Por último nos queda el tráfico *best-effort*, al cuál únicamente se le permite transmitir si se satisface antes cierto rendimiento para dichas colas de menor prioridad.

Con todo ello, para comenzar a desarrollar un modelo con restricciones de forma práctica, podemos centrarnos únicamente en una combinación de los casos isócrono y síncrono en una misma cola frente a otra destinada a tráfico cuya MTU se corresponde con la de Ethernet a fin de simplificar un escenario de pruebas, asemejándose así a una cola *best-effort*. Sin embargo, esta tabla no basta para caracterizar cada uno de los flujos durante la planificación, ya que para ello se requiere conocer, además del periodo de transmisión y el tamaño fijo de las tramas Ethernet, cuáles son la fuente y el destino de la comunicación y el retardo máximo que ésta tolera extremo a extremo.

5.1.2. División temporal: fases e híperperiodo

Como se ha anunciado con anterioridad, cada uno de los flujos puede estar caracterizado por un periodo diferente al del resto, por lo que es necesario dividir el tiempo de un ciclo completo que se repite en intervalos elementales denominados fases o subciclos, tal y como se vio con la Figura 3.1. De esta forma, las diferentes fases conforman lo que hemos denominado híperperiodo, que no es más que el resultado de calcular el MCM al conjunto de periodos de todos los flujos. Con esta división temporal, ya es posible realizar la planificación para todo el conjunto de flujos sin que la periodicidad se vea afectada y favoreciendo el mínimo retardo, pues se pretende realizar la planificación de forma independiente para cada una de estas fases. Por tanto, existirán tantas fases como el cociente resultante de la división de este híperperiodo entre el valor del periodo mínimo del conjunto. Es lógico pensar que, cuanto mayor sea la variabilidad de estos periodos, mayor podrá ser el número de fases a planificar. A esto se le añade que el intervalo de tiempo que comprende cada fase se verá limitada al valor del periodo más pequeño, por lo que a medida que se reduzca éste se podrán planificar un menor número de flujos en cada una. Según el periodo de cada flujo, la planificación se realizará en tantas fases como sea necesario para cumplir con su periodo. En el caso de que los periodos sean múltiplos entre sí (e.g. potencias de dos), todas las fases contendrán al menos una planificación de

tráfico determinista, de lo contrario existirán fases destinadas únicamente al resto de colas como es el caso del *best-effort*. Es muy importante que los tiempos planificados en cada fase para cada nodo se adecuen a los tiempos de transmisión de los nodos previos en la ruta que sigue el flujo a fin de respetar su periodo así como perseguir el mínimo retardo.

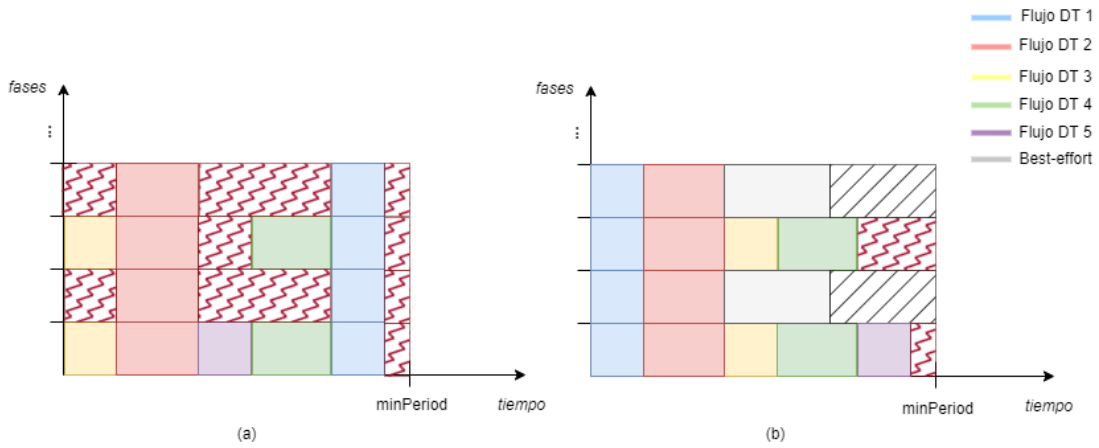


Figura 5.1: Orden de planificación de los flujos en las distintas fases

También es interesante conocer el orden en la planificación que tienen los distintos flujos pues, tal y como se presenta en la Figura 5.1, no se obtienen los mismos resultados en cuanto a la utilización del enlace. Como se puede ver en el primer caso, aparecen rayados en rojo un mayor número de huecos en los que no se puede asignar un tráfico perteneciente a otra cola de menor prioridad o *best-effort* dado que el intervalo de tiempo remanente no es suficiente para la apertura de la puerta de tal cola, principalmente cuando se producen entre flujos de mayor criticidad al limitar aún más dichos espacios, con lo que no se están aprovechando correctamente los recursos de los que se dispone. Recordemos que, para que dicho hueco pueda ser asignado a otra cola, ha de ser tal que pueda albergar al menos una MTU para este tipo de tramas y que cumpla con la banda de guarda para que el tráfico determinista que viene a continuación no se vea afectado. Esta banda de guarda posee el mismo tamaño que la MTU, pues es posible que se esté transmitiendo una trama de tales características al cerrar la puerta de esta cola. En el caso de *best-effort* este tamaño de MTU es de 1542 Bytes, lo que vienen a ser unos 12 μs como se vio en el Apartado 2.2, mientras que este tamaño máximo podrá variar entre colas en función del tipo de tráfico albergado, por lo que conviene tenerlo en cuenta a la hora de clasificarlo. En cambio, en el segundo caso, si se ordenan de menor a mayor periodo, se puede lograr un mayor aprovechamiento de los recursos ya que no existen huecos innecesarios entre flujos deterministas que acaben limitando la asignación de tiempos a otras colas. En este caso es posible que sigan apareciendo algunos huecos a la derecha entre el último flujo planificado y el final de la fase, pero la reducción es notable. Bajo este criterio se reducen también las transiciones del GCL entre las diferentes colas. Por tanto, es necesario que los flujos se ordenen en una lista principalmente en función de su periodo.

Por otro lado, también es necesario otorgarle una mayor prioridad a aquellos flujos cuyo retardo máximo es más pequeño dado que implican también mayor criticidad en el caso de que su periodo sea igual a éste, es decir, el caso isócrono. Además, el hecho de concatenar los tiempos de aquellos flujos que poseen una ruta similar permite evitar los posibles huecos que se puedan producir en los nodos principales, es decir, ordenándolos también a través de un origen a un destino. Finalmente, puede terminar de completarse la

clasificación de lista a través del tamaño de las tramas de estos flujos, pues se plantean los casos en los que son fijos. Es por todo ello por lo que se ha decidido llevar a cabo un sistema de planificación basado en la ordenación de estos flujos a partir de sus restricciones. Por tanto, la clasificación se realiza en base a: 1) periodo de transmisión, 2) retardo máximo tolerado, 3) ID nodo origen, 4) ID nodo destino, 5) tamaño de trama. Por tanto, en este trabajo nos centramos únicamente en una lista ordenada de flujos en base a los criterios anteriores, sin llegar a profundizar en cómo puede afectar cambios de orden concretos para maximizar la respuesta ante determinados flujos ya que esto elevaría exponencialmente la complejidad del sistema y, por tanto, del algoritmo y su tiempo de cómputo como ya se vio en el caso de otros planificadores en el Apartado 3.1.1.

Para comenzar a construir el TAS, pueden considerarse flujos reales de verificación cuyos periodos abarquen desde los 250 microsegundos hasta los 2 milisegundos con requisitos diferentes como son el retardo máximo, el tamaño de trama, etc. Estos flujos quedan por tanto definidos de la misma forma en la que se presentan las comunicaciones isócronas y síncronas de la Tabla 5.1. De esta forma, se ha optado por crear una lista aleatoria de flujos en la que cada uno de ellos adquiere como periodo de transmisión un valor entre 250 μs, 500 μs, 1ms o 2 ms, con lo que se tienen un total de 8 fases según se calcula el MCD (intervalo de fase) y el MCM (intervalo del híperperiodo), tal y como se vio en el Apartado 2.2; pero además su retardo máximo oscile entre los 250 μs del caso isócrono y valores más grandes hasta los 5 ms. Asimismo, el tamaño de sus tramas también puede oscilar en potencias de base dos entre 32 Bytes y 1024 Bytes, según el tipo de comunicación; al igual que el origen y destino se determinará de forma aleatoria entre las diferentes zonas en las que se ubiquen los sensores/actuadores y los controladores. Además, dado que el periodo mínimo se ha establecido 250 μs, será éste el que realmente limite la capacidad de la red para admitir el tráfico síncrono al contar con un menor intervalo de tiempo para la asignación de todos los flujos por cada fase, como se comprobará más adelante. Finalmente, si se normalizan los retardos resultantes de ejecutar la planificación respecto a sus cotas de máximo retardo extremo a extremo hará que una pequeña variación del retardo sobre un flujo con una cota más restrictiva sea más diferenciada que la misma para un flujo con mayor flexibilidad en cuanto a ésta. La intención de ello es recompensar la mejora de los flujos más críticos sobre el resto. Con todo ello, se puede pasar a analizar los diferentes elementos que componen la red industrial objeto de estudio.

5.1.3. Arquitectura de la red. Configuración de los nodos y puertos

Ahora, para poder caracterizar una topología de red TSN sobre la que comenzar a trabajar se ha optado por seguir la jerarquía propuesta en [5], donde existe una deslocalización entre los espacios destinados a los procesos industriales en los que intervienen los sensores y actuadores, los cuales se denominan como celdas y que habitualmente corresponden con diferentes ubicaciones en la infraestructura; y aquellos orientados a la computación de los mensajes recibidos por éstos y la respuesta en consecuencia a ellos (e.g. parada de un proceso, notificación al resto de elementos, reducción de la velocidad, etc.). Con ello, se destacan 3 tipos de comunicaciones:

- ***Controller-to-Controller Communication (C2C)***. Se tratan de comunicaciones establecidas entre controladores destinadas a la coordinación, balanceo y jerarquización en la gestión de los diferentes sensores y actuadores de modo que se comparten mensajes sobre estados y tareas en una red industrial más extensa y dis-

tribuida. Este tipo de comunicaciones pueden producirse a diferentes niveles, como es el caso de la conexión entre PLCs ubicados en diferentes celdas de producción así como éstos con los del *Edge*. Por tanto, suelen relacionarse con comunicaciones más críticas, en las que existe una sincronización continua entre todos ellos.

- ***Controller-to-Device Communication (C2D)***. Son el tipo de comunicaciones más extendido, ya que conecta a los diferentes sensores y actuadores con su controlador PLC correspondiente. De esta forma, pueden tratarse igualmente de conexiones con controladores distribuidos entre las diferentes celdas de producción o bien con aquellos más centralizados.
- ***Device-to-Compute Communication (D2Cmp)***. Este último tipo de comunicaciones son menos frecuentes, pues suelen reservarse a aquellos casos en los que sea necesaria la computación en servidores ubicados en *Cloud*.

Por tanto, para tener un escenario más representativo de una red determinista en la Industria 4.0 se ha optado por la topología de la Figura 5.2, donde se tienen un total de siete nodos enumerados del 0 al 6 y los cuales están conectados entre sí a través de enlaces Ethernet, ofreciendo en este caso velocidades de 1 Gbps. Todos ellos siguen una jerarquía de distribución y acceso como podría ocurrir en un escenario industrial real más simple.

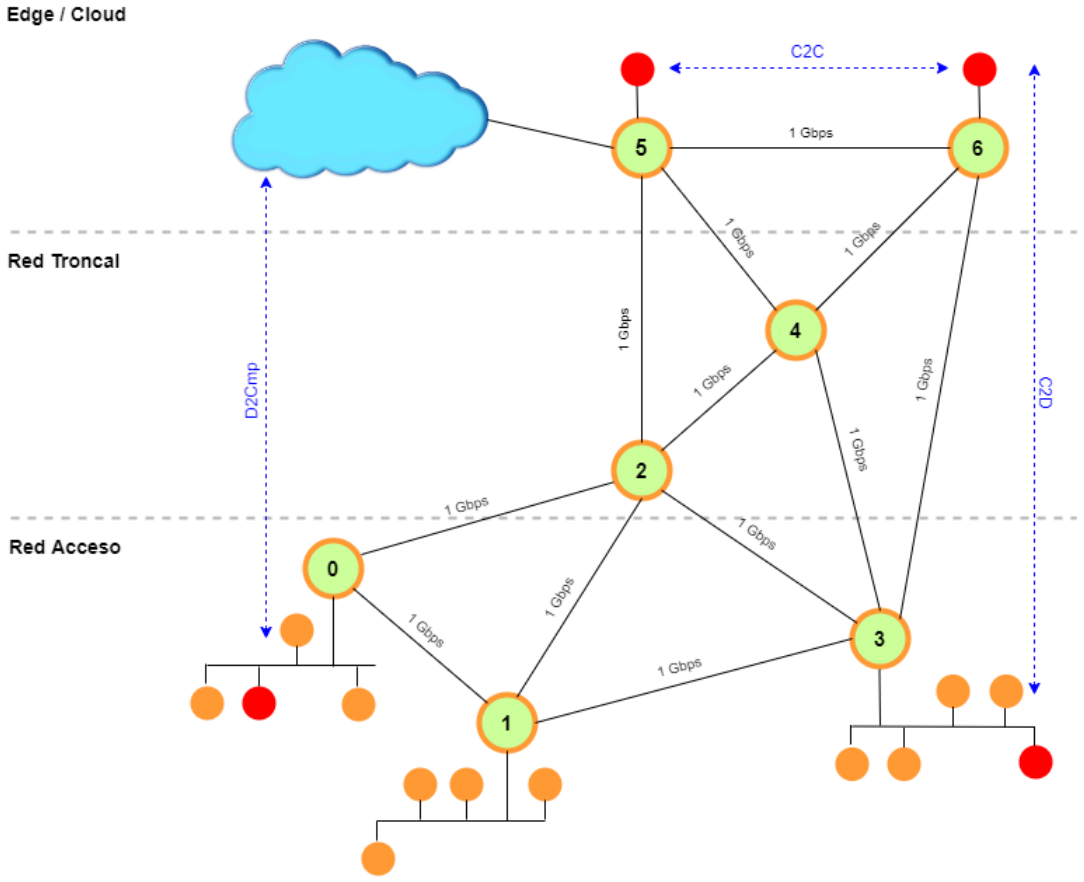


Figura 5.2: Red TSN síncrona desplegada en el entorno de Industria 4.0

Esta topología puede construirse fácilmente a través de la lectura de una única matriz de dimensión $N \times N$, donde N es el número de total de nodos TSN que conforman la red

industrial. Los valores que adquiere cada elemento de esta matriz pueden ser bien 0, lo que significa que no existe un enlace comprendido entre el nodo i hasta el nodo j; o bien el valor de una capacidad, implicando así la existencia de un enlace que comunica a ambos nodos. Nótese que esta matriz implementa enlaces *full-duplex*, con lo que la velocidad en un sentido puede diferir respecto a con la que se transmite en el otro sentido. Sin embargo, estas diferencias no son recomendables dado que pueden generar cuellos de botella que provoquen un mal funcionamiento en el sistema de tiempos de la misma, por lo que se aconseja siempre utilizar una misma capacidad de forma homogénea en toda la red TSN salvo necesidad. En el caso de tratarse de una topología *half-duplex*, estos valores deben coincidir para un mismo enlace, pues los tiempos asignados se replicarán en la planificación de un nodo y en la del vecino con el que conecta tal y como se verá más adelante en el Apartado 5.1.5. Por ejemplo, la matriz que construye la topología anterior es la siguiente, donde el valor de i (fila) corresponde al nodo emisor y j (columna) al nodo receptor:

$$topologyMatrix \rightarrow link_{i,j}(Gbps) = \begin{pmatrix} 0 & 10^9 & 10^9 & 0 & 0 & 0 & 0 \\ 10^9 & 0 & 10^9 & 10^9 & 0 & 0 & 0 \\ 10^9 & 10^9 & 0 & 10^9 & 10^9 & 10^9 & 0 \\ 0 & 10^9 & 10^9 & 0 & 10^9 & 0 & 10^9 \\ 0 & 0 & 10^9 & 10^9 & 0 & 10^9 & 10^9 \\ 0 & 0 & 10^9 & 0 & 10^9 & 0 & 10^9 \\ 0 & 0 & 0 & 10^9 & 10^9 & 10^9 & 0 \end{pmatrix}$$

Nótese que la diagonal es nula, pues no se consideran comunicaciones locales con TSN. De esta forma, se puede crear una clase «nodo» a través de Python que materialice tales equipos de red como objetos y que en ellos se identifique a los distintos vecinos a los que se conecta a través de una ID. Para simplificar este modelo, este valor corresponde al número de nodo representado en el diagrama. Además, cada uno de estos nodos será capaz de conmutar las tramas que llegan a través de sus puertos de entrada con una velocidad determinada, lo cual llevará consigo un retardo de procesamiento hasta que se introduce en la cola correspondiente al puerto de salida. A su vez, en función de los nodos vecinos con los que se conecte, se pueden crear otros tantos objetos asociados a un mismo nodo a partir de una clase «puerto», la cual ya se encargue de administrar aspectos como la planificación a través de las cotas anteriores, los tiempos en cola, los huecos existentes, las bandas de guarda, la capacidad del enlace, etc. Por tanto, este método de construcción de la topología permitirá una gran flexibilidad a la hora de crear nuevas topologías simplemente a través de la edición de dicha matriz, así como de los tiempos de conmutación en cada nodo. Además, pueden plantearse velocidades de acceso para cada uno de los dispositivos hasta el primer nodo en el que son planificados para caracterizar estos retardos.

5.1.4. Espacio de búsqueda y cálculo de posibles rutas

Dado que cada flujo cuenta con una serie de requisitos QoS, se ha decidido implementar también una clase «flujo» en la que se incluyen tales parámetros leídos de la lista ya ordenada en base a los criterios previamente definidos. Entre estos parámetros se encuentran un identificador incremental en el que se establece su posición en dicho orden, el nodo origen, el nodo destino, el retardo máximo, el periodo de transmisión o la longitud de la trama. Además, en este objeto se pueden establecer los tiempos globales de inicio y final de la transmisión resultado de llevar a cabo la planificación del mismo en los diferentes saltos de un vector ruta, de modo que se puede calcular el retardo extremo a extremo simplemente calculando la diferencia entre el inicio y el final en el primer y último nodo,

respectivamente; y añadiendo los tiempos de transmisión y procesamiento en los enlaces de acceso. Para calcular las posibles rutas existentes para un flujo, se ha hecho uso del algoritmo *Depth-First Search (DFS)*, el cual consiste en una búsqueda en profundidad con el fin de ir añadiendo a una lista las posibles rutas que, partiendo desde el nodo origen, conecten con el nodo destino a través de la topología. De este modo, se irán expandiendo todas las posibles ramificaciones que partan desde un mismo nodo sin llegar a producirse bucles entre ellos. En el caso de descubrirse un nuevo nodo éste se añadirá a una lista donde se comprueba primero el último hallado, es decir, se profundizará primero sobre un camino y no a través de las posibles conexiones. Este proceso se realizará con todos los nodos en la lista hasta que quedar vacía, lo que significa que se habrán encontrado todos los posibles caminos. El pseudocódigo empleado para ello es el del Algoritmo 1. De este modo, el espacio de búsqueda será tan amplio como posibles rutas se encuentren para cada uno de los flujos. Cabe a destacar que se pretende disponer del mayor espacio de búsqueda posible con el fin de optimizar el resultado a través de las combinaciones. En cambio, esto elevará el tiempo de computación de nuestro sistema basado en algoritmos genéticos dado que cada gen podría adquirir un valor de un conjunto más amplio, así con un mayor número de generaciones hasta encontrar la solución que maximice el *fitness*. Por ello, se ha decidido limitar este espacio de posibles soluciones para un mismo gen o flujo de modo que primero se clasifique dicho vector en función del número de saltos con un número máximo de nodos por ruta (línea 9) para posteriormente acortarlo a un número n de rutas posibles (líneas 26 a 28).

Algoritmo 1 Búsqueda de posibles rutas para un flujo con DFS

```
 1: src ← flow.src
 2: dst ← flow.dst
 3: paths2check ← {src}
 4: checkedList ← {}
 5: routes ← {}
 6: while paths2check is not Empty do
 7:     path ← paths2check[end]
 8:     paths2check[end] ← {}
 9:     if path.length ≤ flow.maxPathLength then
10:         if dst in path then
11:             routes[end + 1] ← {path}
12:         else
13:             if path not in checkedList then
14:                 checkedList[end + 1] ← {path}
15:                 for node in N do
16:                     if node not in path & topologyMatrix[path[end]][node] > 0 then
17:                         path[end + 1] ← {node}
18:                         paths2check[end + 1] ← {path}
19:                         path[end] ← {}
20:                     end if
21:                 end for
22:             end if
23:         end if
24:     end if
25: end while
26: if 0 < shortenRoutes ≤ routes.length then
27:     routes ← sortbyLength(routes)
28:     routes ← shortestRoutes(routes, shortenRoutes)
29: end if
```

Este algoritmo ha de ejecutarse una única vez para cada tupla de origen y destino con el fin de no repetir el cálculo, trasladándose así para el resto de flujos con las mismas posibles rutas. Esto es muy importante en el caso de encontrarse con una topología más compleja ya que, si se pretenden planificar un volumen importante de flujos, podría elevar el coste computacional al tratar de encontrar todas las rutas posibles. Por tanto, las rutas halladas se tratan únicamente de un vector compuesto por las identidades de los diferentes nodos por los que pasa, tratándose en este caso de valores comprendidos entre el 0 y el 6. Es por este motivo por el que los puertos que se encuentran en la lista del nodo se identifican a través de la ID del nodo vecino con el que conecta, por lo que una vez llega la trama al mismo puede conmutarlo hacia aquél correspondiente al siguiente salto en la ruta. De esta forma, se gestionarán los tiempos asignados a cada flujo en el puerto en función del algoritmo de planificación así como de los tiempos ya asignados a los flujos anteriores en el caso de haberlos.

Una vez dispuesto el espacio de búsqueda para cada uno de los flujos y comprendido el proceso de conmutación en este modelo, se puede pasar ya a la planificación de la ruta correspondiente a cada uno de ellos en la red industrial TSN.

5.1.5. Algoritmo de planificación en TSN síncrono

Para poder llevar a cabo la planificación de un flujo se han de conocer en todo momento los tiempos con los que se han transmitido los anteriores, ya que de ellos dependen los nuevos intervalos de tiempo en los que pueden comprender los siguientes flujos según la ruta designada. De este modo, se ha establecido el valor de una cota inferior o *Lower Bound (LW)*; y de una cota superior o *Upper Bound (UP)*, las cuales pueden verse con la Figura 5.3. Con ello, el planificador ha de seleccionar los tiempos de un flujo de modo que no se solapen con los ya planificados, por lo que han de quedar fuera del intervalo entre ambas cotas a la vez que cumplen con el resto de sus restricciones. De este modo, cada flujo tendrá un tiempo de inicio de la transmisión y un final, caracterizando así su paso por un determinado nodo en función de las características de éste y los enlaces utilizados.

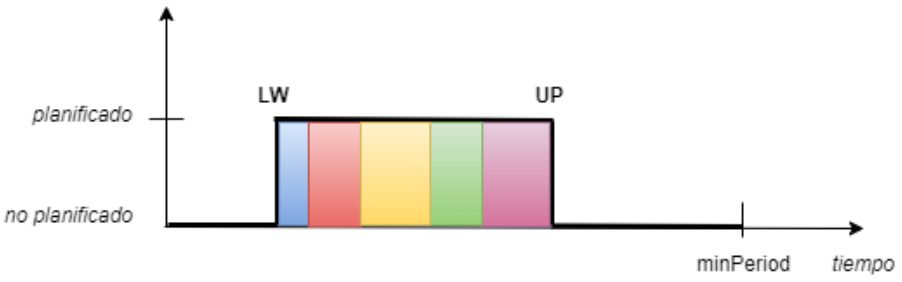


Figura 5.3: Acotación del tiempo planificado para una misma fase

La cota inferior en los nodos desde los que parten los flujos será cero, pues es el primer tiempo establecido para un flujo en la red gracias a la flexibilidad que le da el primer salto ya que se trata en todo momento de aprovechar lo máximo posible el espacio. Sin embargo, puesto que la propia transmisión del flujo acarreará un retardo que se suma al de procesamiento del nodo, en nodos intermedios esta cota inferior podrá ser mayor a cero, según se lleve a cabo todo el proceso de planificación y las características de los flujos con los que se trabaje, principalmente de los primeros que se planifiquen en tales nodos; pero siempre se tendrá el mismo valor de cota inferior para todas las fases dado

que la planificación comienza con los de menor periodo según se ha priorizado. En este caso, dependiendo del tipo de planificación, el hueco que se origina entre el comienzo de la fase y LW podrá utilizarse o no por otro flujo, como veremos más adelante en este Apartado 5.1.5. En cualquiera de los casos, el tiempo que no se destine a este tipo de flujos podrá ser utilizado por el resto de colas, respetando en todo momento el espacio disponible y las bandas de guarda. Por tanto, la transición entre fases no limita a otras colas de prioridad no sensibles al tiempo o el *best-effort* dado que pueden llevar a cabo su transmisión ininterrumpidamente desde la cota superior de una fase hasta la banda de guarda previa a la cota inferior de la siguiente en la que aparezcan flujos planificados.

Dado que se pretende planificar siempre desde la primera fase disponible, en el caso de que no exista espacio suficiente entre la cota superior de la planificación y el límite de la fase para la inserción de un nuevo flujo dentro de la misma es posible planificarla en la siguiente, siempre y cuando el periodo del mismo no coincida con el periodo mínimo ya que en este caso no sería viable la planificación en la ruta designada al no poder cumplir con dicha periodicidad por producirse un desbordamiento. Esto puede verse mejor con la Figura 5.4. Esto es aplicable únicamente al primer nodo ya que, de lo contrario, de realizarse este cambio de fase en un nodo intermedio, una trama estaría siendo retenida en cola desde su llegada al puerto hasta la siguiente fase durante la cual se transmiten otras tramas con tiempo de llegada posterior, lo cual rompería con el esquema de colas FIFO planteado. En este esquema de cotas no es posible la transmisión de un flujo entre dos fases, es decir, con tiempo de inicio de la transmisión en una fase y con el final de ésta en la siguiente, pues esto provocaría una incompatibilidad de cotas dentro de la misma impidiendo así ajustes para poder minimizar el retardo.

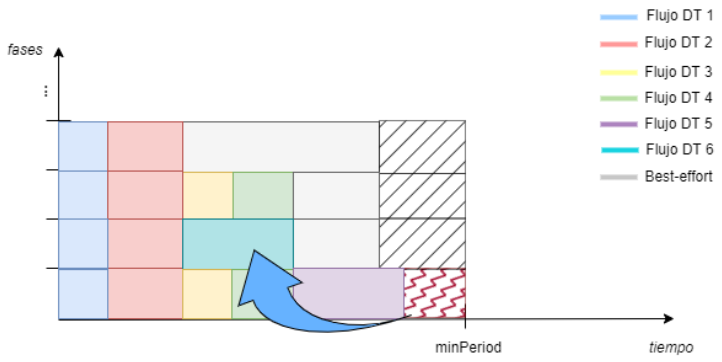


Figura 5.4: Cambio de fase en la planificación de un flujo por desbordamiento

En el caso en el que se adoptara un método de discretización de este tiempo de fase como el visto en el Apartado 3.1.1 para la planificación en pequeñas ranuras con el menor tamaño de trama posible de 32 Bytes, puesto que el retardo que introduce cada salto i para un flujo m en su paso por la red se mide en la Ecuación 5.1 como la suma del retardo de procesamiento del nodo (t_{proc}) más el tiempo de espera en cola (t_{queue}), el de transmisión (t_{tx}) y el de propagación (t_{prop}) tal y como se vio en la Figura 3.2; podría dar lugar a un incremento del retardo al tener que permanecer una trama en cola hasta el comienzo de la siguiente ranura disponible. Por otro lado, aunque las ranuras que quedaran libres se podrían utilizar para el resto de tráfico cumpliendo las anteriores premisas, podría llegar a reducirse el nivel de utilización de un enlace en el caso en que quedaran ranuras individuales ya que no podrían albergar siquiera las bandas de guardas. Sin embargo, el mayor problema de este sistema es que no sería capaz de ajustar una banda de guarda

a la variabilidad introducida por las NFV como se vio en el Apartado 3.2, por lo que sería bastante ineficiente. Respecto a esta ecuación, podría considerarse un tiempo fijo (e.g. 10 μs) para el procesamiento de tramas dada su escasa variabilidad entre tamaños y despreciarse el tiempo de propagación para simplificar el problema de modo que tan solo se tengan en consideración éste junto con el tiempo de transmisión en función de la capacidad del enlace que conecta con el siguiente nodo en el camino y el tamaño de la trama a transmitir (fr^m_{size}); y el tiempo de espera en cola.

$$t^i = t_{proc} + t_{queue} + t_{tx} + t_{prop} = fr^m_{size} \cdot proc^i_{speed} + \left(t^{m-1}_{end} - t^m_0\right) + \frac{fr^m_{size}}{port^{i,i+1}_{speed}} + \frac{d^{i,i+1}_{link}}{v_{prop}} \tag{5.1}$$

Con esto, una vez llegue al nodo una trama correspondiente a un determinado flujo, se añadirá este tiempo de procesamiento al cómputo general del retardo. De este modo, se puede establecer un instante t_0 de llegada al puerto de salida a partir del cual el algoritmo de planificación pueda ubicar los tiempos de transmisión en su reserva. Dicho instante t_0 es el resultado de sumar este tiempo de conmutación al tiempo de final de transmisión del nodo anterior. Por tanto, jamás podrá reservar un intervalo de tiempo para la transmisión de un flujo en ninguna fase de forma que se produzca ésta anticipadamente a su llegada. En cambio, existe una excepción cuando se trata del primer nodo en la ruta ya que previamente no ha sido transmitido por ningún otro mas que por el dispositivo. En este caso su valor será de cero, por lo que podrá planificarse para cualquier fase y en cualquier instante de ésta siempre que se respeten las cotas. Se ha decidido por tanto excluir los tiempos de transmisión y recepción de los dispositivos finales, así comprendiendo la planificación de sus tiempos únicamente desde su ingreso en la red TSN hasta su salida. Con ello, al retardo extremo a extremo habrá que añadirle al retardo introducido en la ruta de K nodos tanto los tiempos de procesamiento de los nodos limítrofes ($t_{proc_{0,K-1}}$) como los tiempos de transmisión a través de los enlaces de acceso ($t_{acc_{src,dst}}$) y que conectan dichos dispositivos con la red industrial, así resultando el retardo extremo a extremo total de la Ecuación 5.2.

$$delay_{total} = \sum_{i=0}^{K-2} t^i + t_{proc_0} + t_{proc_{K-1}} + t_{tx_{acc-src}} + t_{rx_{acc-dst}} \tag{5.2}$$

Sin embargo, este instante t_0 se trata de un valor temporal global obtenido a partir de los tiempos del flujo y que comprende todo un híperperiodo, es decir, no distingue de fases, por lo que se ha de calcular a cuál de ellas pertenece. Esto se hace a través del cociente resultante de dividir dicho tiempo entre el valor del periodo mínimo y de su módulo, obteniendo así la fase en la que se encuentra como el instante de llegada al puerto dentro de la misma fase, respectivamente. Esto no será necesario en el último nodo ya que, para simplificar, cada dispositivo receptor se conecta a un único puerto, por lo que ya solamente existe una única cola a la que llega el tráfico independientemente de su prioridad.

De esta forma, se han reconocido hasta tres posibles casos que pueden llegar a acontecer en la planificación de un flujo en una fase concreta sobre un puerto ante la llegada de una trama a éste:

- **Caso 1**. $[t_0 > UP]$: Ocurre cuando la llegada de la trama se produce después de la planificación de todos los flujos anteriores. De este modo, no se encuentra con otras tramas en cola, por lo que se puede inicializar el intervalo de transmisión desde el

mismo instante en el que llega, es decir, t_0. Esto será posible siempre y cuando no se rebase el límite que impone el fin de fase. Sin embargo, habrá que guardar el intervalo de tiempo comprendido entre UP y t_0 como hueco ya que durante éste no se estará transmitiendo tráfico sensible al tiempo correspondiente a esta cola.

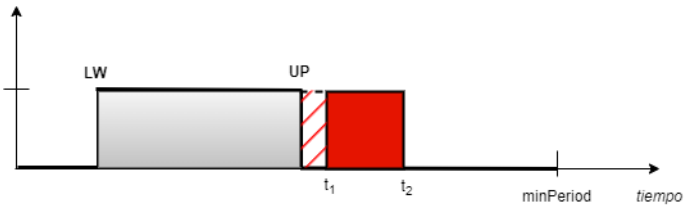


Figura 5.5: Caso 1 en la planificación de un flujo

- **Caso 2**. $[t_0 + t_{tx} < LW]$: Corresponde cuando la trama llega antes que el resto de los flujos ya planificados de forma que su tamaño en Bytes así como la capacidad del enlace permiten que ésta pueda transmitirse sin llegar al límite inferior, es decir, el marcado por LW. De no ser así, su planificación no sería posible salvo el caso del primer salto, donde tendría que retrasarse su llegada para transmitirse justo después de UP. Con ello, tampoco aparecerá un tiempo de espera en cola.

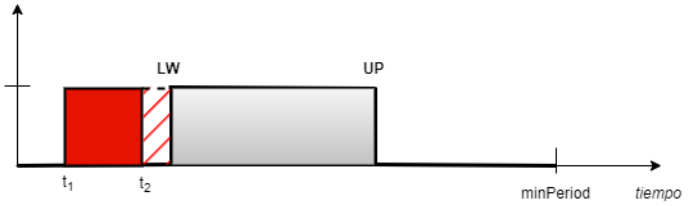


Figura 5.6: Caso 2 en la planificación de un flujo

- **Caso 3**. $[t_0 \leq UP]$: Este caso engloba al anterior cuando no puede transmitirse antes de la cota inferior y cuando la llegada de la trama se produce entre esta LW y UP, debiéndose transmitir también justo después de UP. Esto, de nuevo, será posible siempre y cuando no se rebase el límite de la fase, aunque podrá introducir retardo por espera en cola. Además, el tiempo de llegada debe ser mayor al del último flujo que establece el UP de modo que se respete el esquema de colas FIFO.

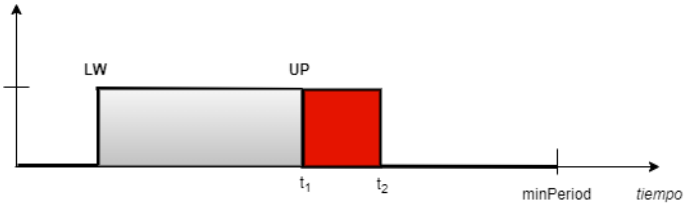


Figura 5.7: Caso 3 en la planificación de un flujo

Todo esto puede traducirse al pseudocódigo del Algoritmo 2, donde se presenta de una forma más básica el procedimiento para la planificación de un flujo en un puerto determinado en función del caso que se presente con la llegada de una trama. Como se irá viendo más adelante, el algoritmo ganará una mayor complejidad.

Algoritmo 2 Planificación de un flujo en un nodo i de la ruta

```
1: for port in node do
2:     if port.ID is path[i + 1] then
3:         t_0 ← flow.t_2[i − 1] + t_proc_i
4:         phase ← t_0 / minPeriod
5:         t_0 ← t_0 (mod minPeriod)
6:         scheduled ← 0
7:         j ← 0
8:         while (phase + j) < numPhases do
9:             if j · minPeriod (mod flow.T_tx) = 0 & scheduled < numPhases then
10:                if port.UP[phase + j] < t_0 then                          ▷ Case 1
11:                    t_1[phase + j] ← t_0
12:                    t_2[phase + j] ← t_0 + t_tx
13:                    if t_2[phase + j] < minPeriod then
14:                        port.gaps[phase + j][end + 1] ← {port.UP[phase + j], t_1[phase + j]}
15:                        port.UP[phase + j] ← t_2[phase + j]
16:                        scheduled ← scheduled + 1
17:                        j ← j + 1
18:                    else if not scheduled & i = 0 & flow.T_tx ≠ minPeriod then
19:                        phase ← phase + 1
20:                        t_0 ← 0
21:                    else
22:                        Flow could not be scheduled
23:                    end if
24:                else if t_0 + t_tx < port.LW[phase + j] then              ▷ Case 2
25:                    t_1[phase + j] ← t_0
26:                    t_2[phase + j] ← t_0 + t_tx
27:                    port.gaps[phase + j][end + 1] ← {t_2[phase + j], port.LW[phase + j]}
28:                    port.LW[phase + j] ← t_1[phase + j]
29:                    scheduled ← scheduled + 1
30:                    j ← j + 1
31:                else                                                      ▷ Case 3
32:                    t_1[phase + j] ← port.UP[phase + j]
33:                    t_2[phase + j] ← t_1[phase + j] + t_tx
34:                    if t_2[phase + j] < minPeriod then
35:                        port.UP[phase + j] ← t_2[phase + j]
36:                        scheduled ← scheduled + 1
37:                        j ← j + 1
38:                    else if not scheduled & i = 0 & flow.T_tx ≠ minPeriod then
39:                        phase ← phase + 1
40:                        t_0 ← 0
41:                    else
42:                        Flow could not be scheduled
43:                    end if
44:                end if
45:            else
46:                j ← j + 1
47:            end if
48:        end while
49:        break
50:    end if
51: end for
52: flow.t_1[i] ← t_1
53: flow.t_2[i] ← t_2
```

Como puede observarse, esto se realiza a través de la búsqueda del puerto cuyo identificador coincide con el del nodo que brinda el siguiente salto en la ruta y con la periodicidad requerida, con lo que se tiene un bucle que repite la planificación para cada fase en la que corresponde el periodo de transmisión del flujo, es decir, cuando el resultado de aplicar el módulo del periodo al inicio de la fase en la que se encuentra es cero. Por ejemplo, en el caso que abordamos, si el periodo mínimo es de 250 μs, el proceso se llevará a cabo cuando el módulo de los múltiplos j de éste entre el periodo de transmisión sea cero, es decir, en todas las ocho fases; mientras que en el caso de 1 ms solamente se realizará cuando j sea 0 ó 4. De este modo, el algoritmo planificador se ejecutará de la misma forma en cada uno de los nodos, a excepción del último salto dado que en éste no se planificará al conectar directamente con el destino como venimos explicando. Para cada fase se actualizarán las cotas del puerto de modo que los siguientes flujos a planificar, de haberlos, puedan considerar dicha referencia para evitar las colisiones. Del mismo modo, en caso de producirse un hueco, añade su intervalo a una lista del puerto y la fase para ser contabilizado. A su vez, los tiempos de inicio y final de la transmisión del flujo en el nodo también se van guardando en un vector para finalmente añadirlos al propio campo del objeto flujo con el fin de tenerlos en cuenta en los siguientes saltos a través del parámetro t_0; así como a la hora de medir el retardo extremo a extremo. Además, con las líneas 18-20 y 38-40, se presenta el caso anteriormente descrito con la Figura 5.4 en el que un flujo no puede ser designado a una fase en su nodo inicial debido al desbordamiento producido al ser el tiempo final de transmisión (t_2) mayor que el establecido como el límite de la fase o periodo mínimo.

En cambio, una limitación importante de este sistema es que, de haberse planificado un flujo en una de las fases, en el caso en que se produzca desbordamiento en alguno de los nodos siguientes al primer salto en la ruta no será posible la planificación y automáticamente se descartará dicha solución. Resulta lógico pensar que una posible solución a esto recae en planificar el flujo en la siguiente fase, tal y como se vio con la Figura 5.4, donde se encontrará con un rango de tiempo más amplio en el que poder comenzar su transmisión tal y como se ha planteado el sistema. Para ello es necesario sustituir en el Algoritmo 2 la condición de las líneas 13 y 34 por un margen en su inicio en la planificación del primer nodo con el objetivo de que pueda ser desplazado a la siguiente fase en el caso de que no pueda cumplir con más de un salto, es decir, se puede establecer por ejemplo que en el nodo inicial deba cumplir que el espacio restante hasta el final de la fase sea de un 10 % del total del periodo mínimo, en este caso tratándose el margen de 25 μs que permitan a la trama llegar hasta su destino. De este modo, se estará limitando más la cabida de flujos en esta fase, pero por el contrario permitirá evitar descartar directamente la solución mientras otras fases quedan vacías. Además, si se introduce un margen adecuado, puede ser aprovechado para el envío de al menos una trama correspondiente a la cola *best-effort*, como en este caso los 25 μs tal y como se ve con la Figura 5.8, donde se pueden albergar esos 1542 Bytes de trama Ethernet más la banda de guarda del mismo tamaño, en el caso de velocidades de 1 Gbps. Este margen se tornará nulo para este flujo en el caso de los nodos siguientes al primer salto ya que ahora sí podrán llegar hasta el límite de fase, tratando de no rebasarlo para evitar que la solución sea descartada. Sin embargo, esto no siempre será así ya que, en el caso de tramas más grandes o con un mayor número de saltos podrían no llegar a su destino y sería necesario un margen más amplio, algo que, como se verá en el siguiente subapartado, se podría agudizar en el caso de retrasarse sus tiempos. De momento, se mantendrá un valor fijo del 10 % para mayor simplicidad, aunque podría tratarse de un parámetro más del cromosoma a fin de hallar un valor porcentual que facilite también esta optimización.

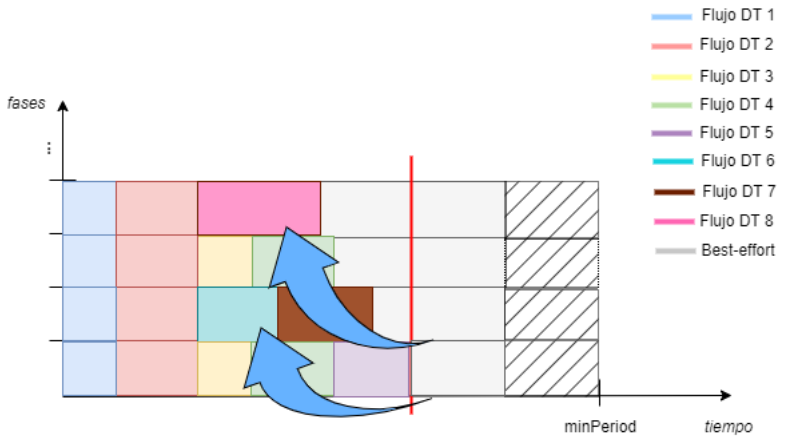


Figura 5.8: Margen de fases en primer nodo para evitar el desbordamiento en los siguientes

Retraso de tiempos en la ruta *zero-wait*

Por tanto, este sistema implica que el intervalo de transmisión no siempre comenzará en el instante t_0, pues es posible que deba permanecer en cola hasta que se terminen de transmitir las tramas planificadas con anterioridad. Además, dado que estamos realizando una planificación basada en la prioridad y no tanto de forma temporal, es posible que se produzca una discordancia entre en el tiempo de llegada de una trama correspondiente a un flujo y las ya planificadas pertenecientes a otros flujos pues, debido a circunstancias como el tamaño de dicha trama o la ruta hasta tal nodo, la última podría llegar antes que el resto, rompiendo así con el sistema de colas FIFO como ya se ha mencionado. Se puede solucionar este problema implementando un sistema en el que no exista tiempo de espera en cola, lo que denominamos *zero-wait*. Para ello es necesario retrasar los tiempos de transmisión en los nodos previos desde el inicio de la ruta hasta el nodo actual que se está planificando de modo que pueda comenzar la transmisión conforme la trama llegue al puerto, inmediatamente después de terminar de transmitirse el flujo anterior. Dicho retraso corresponderá con la diferencia temporal que existe entre el instante t_0 hasta la cota superior del puerto, es decir, el tiempo de final de transmisión del flujo anteriormente planificado en dicho puerto. Por tanto, se estarán modificando los tiempos del flujo en cada nodo, pero también los de las cotas definidas en el puerto. Con esto aparecerá nuevos huecos en estos nodos entre el final de la transmisión del flujo anterior y el nuevo tiempo designado para la del flujo actual; o también llegar a extender aún más los ya existentes en el caso de que se produzca más de un retraso para el mismo flujo o haya ocurrido el Caso 1 de llegada en alguno de los nodos que conforman la ruta, por lo que será necesario comprobar primero estos tiempos. En cambio, el hecho de retrasar estos tiempos permite mantener el retardo sin llegar a incrementarlo dado que la transmisión del dispositivo se adaptará también a los tiempos en el primer nodo de modo que finalice en el instante t_0. Esto puede visualizarse con la Figura 5.9 en un caso en el que participan hasta tres nodos.

En los Casos 1 y 3 con los que una trama llega a los nodos, este retraso se realiza de forma sencilla, pues el único límite que tiene para desplazarse es el final de la fase. Sin embargo, en el Caso 2 esto no es así, pues está más limitado por LW. De este modo, en el caso de tener que retrasarse un flujo en un nodo en el que se haya planificado antes que el resto, únicamente contará con el hueco que existe entre su final de transmisión y LW,

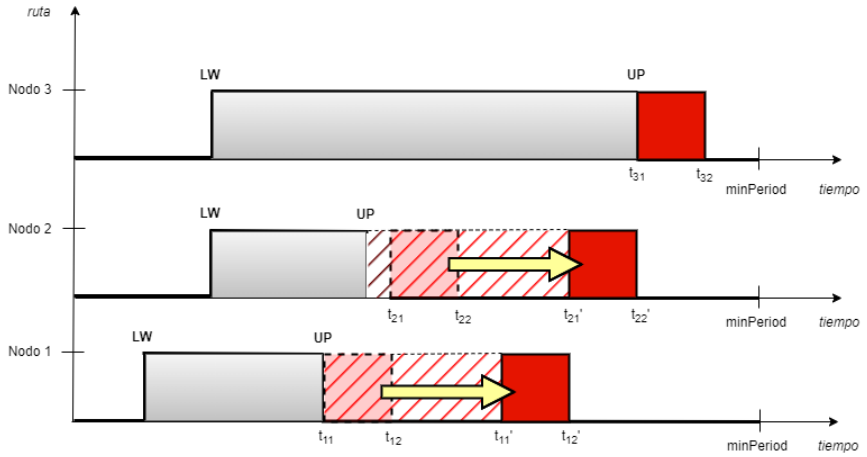


Figura 5.9: Resultado de retrasar un flujo a lo largo de la ruta en *zero-wait*

por lo que si este retraso es superior no podrá planificarse de esta forma al colisionar con otros flujos. Por este motivo, pese a haber considerado este caso y haberlo programado, se ha optado por no hacer uso de él en la planificación ya que el hueco existente entre el inicio de la fase y LW podría ser aprovechado por otra cola y el flujo planificarse en esta u otra fase, según proceda. Además, como detalle, cabía la posibilidad de realizar una pequeña compresión de modo que el instante t_2 coincidiera con el de LW con el fin de evitar el hueco, pero definitivamente esto no funcionaría con este nuevo esquema. Por tanto, es preferible utilizar tan solo con los Casos 1 y 3 del Algoritmo 2 para la planificación de flujos, pues de este modo es posible tener en todo momento un t_0 igual o mayor a UP. Con ello, se añade el Algoritmo 3 para el retraso del flujo a lo largo de los nodos. Cabe a mencionar que este algoritmo había sido diseñado en su origen también para el Caso 2 y con el objetivo de resolver la discordancia con las colas FIFO, por lo que se había establecido un retraso a medida que se progresaba en el camino de forma inversa hasta llegar al primer nodo en la ruta. De esta forma, era posible aplicar este retraso únicamente en los últimos nodos sin tener que completar este procedimiento en toda una ruta, principalmente cuando se producía el Caso 2 sin poder desplazar los tiempos debido a colisiones con tiempos posteriores a LW. Sin embargo, para ello era necesario cumplir con el retraso en al menos un nodo, es decir, en el último planificado, pero esto elevaría el retardo extremo a extremo ya que los tiempos de transmisión se mantendrían mientras en algunos nodos se retrasarían, con lo que se decidió descartar dicho caso e implementar la medida *zero-wait*. Respecto a este nuevo algoritmo se tiene por tanto que, en el caso de coincidir el UP del puerto con el t_2 del flujo a planificar (Casos 1 y 3), los tiempos se incrementará en *dif* (valor diferencia entre t_0 y UP) siempre que no sobrepase el límite de fase; mientras que en el caso de que coincida el LW del puerto con el t_2 del flujo (Caso 2) aumentará *dif* solo si el hueco que exista a continuación de t_2 es suficiente. Nótese que, en el caso de no ser posible alguno de ellos en el primer nodo, la planificación no será posible para el flujo, por lo que la solución no será apta y se descartará. Por tanto, para el Caso 2 se tendría un gran número de situaciones en el que no se cumple este requisito, lo que hace que el sistema descarte un mayor número de soluciones al usar la zona de la fase que queda a la izquierda del LW. De este modo, empleándose solamente los Casos 1 y 3, si el primer flujo planificado en la fase llega en un instante superior al de inicio de fase, se creará un hueco de tales características para ser posteriormente contabilizado. Con todo esto, el Caso 3 será el único que invocará a la función planteada con este último algoritmo.

Algoritmo 3 Retraso del flujo a lo largo de la red TSN en paradigma *zero-wait*

```
1: dif ← UP − t_0
2: for node in reversePath do
3:     for port in node do
4:         if port.ID is reversePath[i + 1] then
5:             isPossible ← 0
6:             for phase in port do
7:                 if flow.ID in port.scheduledFlows[phase] then
8:                     if flow.t_2[node] = port.UP[phase] then                ▷ Cases 1 and 3
9:                         if port.UP[phase] + dif < minPeriod then
10:                            port.gap[phase] ← {flow.t_1, flow.t_1 + dif}
11:                            flow.t_1[node] ← flow.t_1[node] + dif
12:                            flow.t_2[node] ← flow.t_2[node] + dif
13:                            ports.UP[phase] ← port.UP[phase] + dif
14:                            isPossible ← 1
15:                        else
16:                            Flow could not be scheduled
17:                        end if
18:                    else if flow.t_1[node] = port.LW[phase] then           ▷ Case 2
19:                        for gap in port.gaps[phase] do
20:                            if gap.start = flow.t_2[node] & (gap.end − gap.start ≥ dif) then
21:                                gap.start ← gap.start + dif
22:                                flow.t_1[node] ← flow.t_1[node] + dif
23:                                flow.t_2[node] ← flow.t_2[node] + dif
24:                                isPossible ← 1
25:                                break
26:                            end if
27:                        end for
28:                        if not isPossible then
29:                            Flow could not be scheduled
30:                        end if
31:                    end if
32:                end if
33:            end for
34:        end if
35:    end for
36: end for
37: t_0 ← flow.t_2[i − 1] + t_proc_i    ▷ New t_0 value is calculated, so Case 3 is performed (t_0 = UP)
```

Para que todo ello funcione correctamente, es necesario que cada vez que se genera un hueco éste se combine con aquellos cuyos límites del intervalo coincidan de forma que tanto en el algoritmo de retraso de flujos (línea 20) como en la evaluación de huecos en la función *fitness* considere su duración real, pues de lo contrario el sistema en ocasiones podría estar dando lugar a fallos cuando no debería de haberlos en el Caso 1 y evaluando de forma incorrecta las soluciones, pues recordemos que en ella se mide la capacidad del hueco de albergar tramas MTU de Ethernet y considerando las bandas de guarda antes de la transmisión de un flujo, así invalidando por completo una solución que podría llegar a ser óptima.

En cambio, como se ha visto con la Figura 5.8, en el caso de retrasarse los flujos esto podría dar lugar a rebasar el límite, con lo que será más fácil que se produzca tal desbordamiento en los nodos correspondientes a los posteriores saltos. Por tanto, se ha desarrollado un planificador que se adecua bastante a las características de TSN pero que,

sin embargo, dista de ser la mejor solución en el caso de contar con un volumen muy denso de flujos, aunque puede tratarse como una sólida base para hacerlo más «inteligente».

Por otro lado, se ha comprobado que esto potencialmente ocurre en los nodos intermedios por los que circula un mayor volumen de flujos desde diferentes nodos de acceso mientras que la planificación de éstos últimos queda más estancada con los flujos de aquellos dispositivos a los que se conecta. No obstante, esto también se produce a consecuencia de los tamaños de las tramas pues, cuando dos flujos que siguen la misma ruta, si uno de mayor tamaño precede a otro con menor número de Bytes, dado que el tiempo de transmisión es menor en este último, tendrá que retrasarse hasta hacer coincidir su llegada con el final de la transmisión de la trama más grande y con ello se generará un hueco debido a este sistema de orden de flujos. Esto ocurrirá siempre al darle mayor importancia a los periodos, aunque gracias a ello se evitará en mayor medida dado que no siempre ocurrirá como en el caso de la Figura 5.1. Es principalmente esta característica la que hace al sistema algo menos intuitivo dado que depende directamente del tamaño de los huecos, aunque es razonable que una solución sea peor que otra cuando aparecen un mayor número de huecos más pequeños. De esta forma, el algoritmo genético tratará de encontrar la mejor combinación de rutas posible con el fin de minimizar tanto el retardo como los huecos que no pueden ser utilizados por el resto de colas, adaptándose siempre a las reglas del algoritmo planificador descritas hasta ahora. Nótese que, gracias al sistema de *zero-wait*, podremos mantener unos umbrales de retardo relativamente bajos, haciendo depender a éste únicamente del número de saltos y no tanto de la planificación o el orden de los flujos en un mismo nodo, lo que favorece un mayor control sobre los tiempos.

5.1.6. Consideraciones adicionales: *Half-duplex Vs. Full-duplex*

Como en toda red Ethernet, las ventajas de usar un esquema *full-duplex* son claras, pues permite ofrecer un mayor ancho de banda en los enlaces al poder establecer conexiones de más de un cable físico de mayor categoría en cada sentido así como en este caso evitar colisiones entre ambos, optimizando más la utilización de los recursos. En cambio, cabe la posibilidad de emplear un esquema *half-duplex* en el que se favorezca un rendimiento similar con un menor coste de despliegue cuando se tratan de topologías más complejas en las que se requiere una mayor inversión en el cableado, con un tráfico más localizado y con transmisiones que se producen principalmente en un único sentido (e.g. enlace ascendente) y de forma casi excepcional. De no ser así, las fases planificadas podrán desbordarse fácilmente debido al número de saltos que puede haber desde un punto u otro de la red hasta el mismo nodo, lo que implica tiempos de llegada muy distanciados entre sí y la aparición de huecos muy extensos. Esto puede agravarse aún más con la metodología *zero-wait* anteriormente descrita, así pudiendo albergar un menor volumen de tráfico sensibles al tiempo. Programar esto en nuestro modelo es algo trivial, pues basta con aplicar la reserva de los tiempos para un flujo con los mismos comandos de un puerto j en un nodo i anteriormente mostrados y replicarlos en el puerto i del nodo vecino j en las mismas fases. De esta forma, basta con añadir como parámetro de ejecución un booleano con el que poder ejecutar este caso cuando se active. Para ello, es necesario que las capacidades de la matriz de topología sean iguales en ambos sentidos, pues los tiempos de transmisión dependen también de dichas capacidades como se vio con la Ecuación 5.1 y se trata de un mismo enlace, por lo que de ser diferentes se generarían incongruencias dentro de éste. Sin embargo, dado que en esta topología se pueden producir comunicaciones bidireccionales entre sensores y controladores pero también entre controladores y actuadores o entre va-

rios controladores desde distintas secciones del espacio industrial (véase Apartado 5.1.3), es necesario implementar un sistema de comunicaciones *full-duplex* entre nodos TSN. Por tanto, a partir de ahora nos centraremos únicamente en el caso de *full-duplex*

5.1.7. Resultados ofrecidos por el modelo TSN síncrono

Finalmente, constituido nuestro modelo completo en Python, se puede pasar a comprobar el resultado de la planificación síncrona sobre la topología de red determinista TSN implementada. Para ello, probamos inicialmente con un caso en el que se pretende planificar un total de 20 flujos y cuyas características (periodo, retardo máximo, origen y destino, tamaño, etc.) han sido generadas de forma coherente a dicha topología y a los tiempos descritos con la Tabla 5.1. Las características de estos flujos ya ordenados y planificados pueden verse a continuación en la Tabla 5.2. Las rutas se han designado bajo el propio criterio. Lo más relevante que debe devolver el modelo de planificación como respuesta al controlador de red CNC son los tiempos designados a cada uno de los flujos a lo largo de los diferentes nodos que componen su ruta, ya que a partir de ellos definirá también los tiempos de transmisión para cada uno de los dispositivos a través del CUC a partir del tiempo inicial planificado en el primer salto; así como de los resultados de retardo extremo a extremo para cada uno de ellos y de utilización de los enlaces a evaluar. Además, con ello se podrán definir los tiempos para cada una de las colas a través del GCL.

ID	nodo *src*	nodo *dst*	T_{tx}	$Delay_{max}$	*tamaño*	Ruta	$Delay_{e2e}$
#0	0	3	250 μs	250 μs	128 Bytes	0-2-3	34.096 μs
#1	0	6	250 μs	250 μs	128 Bytes	0-1-3-6	45.12 μs
#2	1	3	250 μs	250 μs	64 Bytes	1-3	21.536 μs
#3	6	1	250 μs	250 μs	32 Bytes	6-3-1	31.024 μs
#4	6	5	250 μs	250 μs	128 Bytes	6-5	23.072 μs
#5	3	0	500 μs	250 μs	64 Bytes	3-1-0	32.048 μs
#6	1	0	500 μs	750 μs	512 Bytes	1-0	32.288 μs
#7	3	6	500 μs	750 μs	32 Bytes	3-6	20.768 μs
#8	3	1	500 μs	500 μs	512 Bytes	3-1	32.288 μs
#9	3	1	500 μs	500 μs	512 Bytes	3-1	32.288 μs
#10	3	1	500 μs	500 μs	1024 Bytes	3-1	44.576 μs
#11	5	3	1 ms	750 μs	64 Bytes	5-2-3	32.048 μs
#12	5	0	1 ms	1 ms	1024 Bytes	5-2-0	62.768 μs
#13	1	5	1 ms	2 ms	256 Bytes	1-2-5	38.192 μs
#14	0	1	1 ms	5 ms	32 Bytes	0-1	20.768 μs
#15	1	0	2 ms	250 μs	512 Bytes	1-0	32.388 μs
#16	3	0	2 ms	500 μs	512 Bytes	3-2-0	46.384 μs
#17	1	5	2 ms	1 ms	64 Bytes	1-2-5	32.048 μs
#18	0	5	2 ms	2 ms	256 Bytes	0-2-5	38.192 μs
#19	1	3	2 ms	5 ms	512 Bytes	1-3	32.288 μs

Tabla 5.2: Características de flujos planificados y retardo E2E conseguido

Comprobando los resultados de esta tabla, el retardo logrado para cada uno de los flujos es en este caso el menor posible ya que las rutas que siguen los flujos son las de menor número de saltos según la ubicación de los nodos que comunican a ambos extremos a través de la red TSN y, dado que se hace uso del esquema *zero-wait*, no hay tiempos de espera en cola, por lo que dicho retardo depende principalmente del tiempo de conmutación junto con el de transmisión a través del enlace y que se ve reflejado en la Figura 5.10, donde puede verse de forma gráfica la planificación de tiempos entre las diferentes fases de estos flujos enumerados en la leyenda, en este caso en el enlace que conecta los nodos 3 y 1.

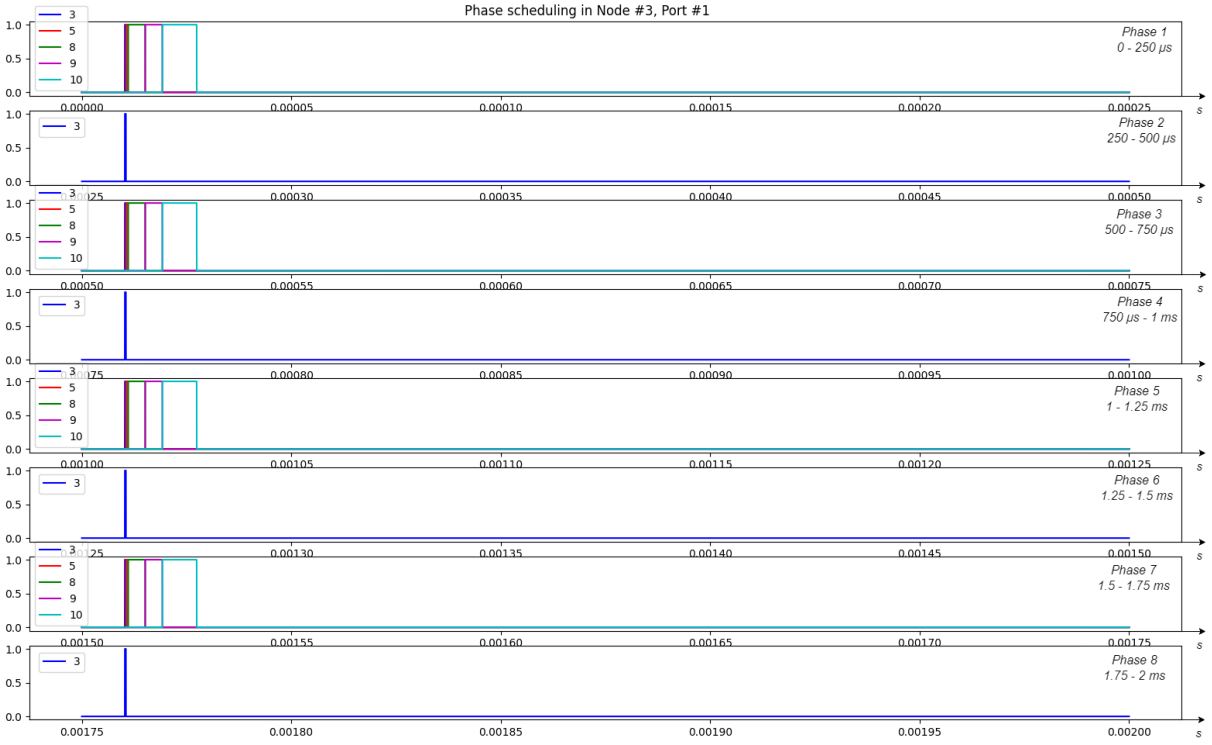


Figura 5.10: Configuración de 20 flujos en un enlace

Como puede observarse, en este enlace se planifican cinco de estos flujos, de los cuales únicamente el flujo 3 posee un periodo inferior que coincide con el periodo mínimo de 250 μs, por lo que se repiten sus tiempos en todas las fases. Nótese que cada fase comprende estos periodos de 250 μs, aunque en la gráfica aparecen ya representados los tiempos globales del híperperiodo de 2 ms. El resto posee un periodo dos veces superior de 500μs, con lo que tan solo se repite en fases alternas. Además, dado que el flujo 3 tiene un periodo menor, es el primero en planificarse según el método de ordenación utilizado por lo que, al corresponderse este enlace con el segundo salto de la ruta seguida por éste, su tiempo de inicio de transmisión se verá desplazado hasta aproximadamente los 10.256 μs desde el comienzo de la fase, como puede apreciarse mejor con las dos últimas fases en la Figura 5.11, tiempo que coincide con el instante final de la transmisión desde el nodo 6 (pues se trata de tramas de tan solo 32 Bytes) más el retardo introducido por el proceso de conmutación del nodo y que se ha establecido en 10 μs, con lo que también lo harán los tiempos de los flujos que vayan a continuación. De este modo, se almacena como un hueco el intervalo que comprende desde el inicio de la fase hasta el inicio de la transmisión de este flujo para luego ser computado en la evaluación. Por tanto, este tiempo de inicio de la transmisión del primer flujo coincide con el de LW, mientras que el de UP se fija con el final

de la última transmisión, por lo que el tamaño de este hueco no afectará a la evaluación. Sin embargo, este hueco es un tanto especial, ya que no se mide de forma individual, sino que se añade al tiempo restante desde el UP hasta el final de la fase anterior ya que recordemos son continuos en el tiempo. Con todo ello, el retardo extremo a extremo depende de dicho número de saltos así como del tamaño de las tramas y la velocidad de transmisión de los enlaces, que en este caso es la misma para todos, 1 Gbps. También nos encontramos con el caso en el que un dispositivo podría realizar una ráfaga de dos tramas seguidas a través de los flujos 8 y 9 ya que ambos poseen las mismas características.

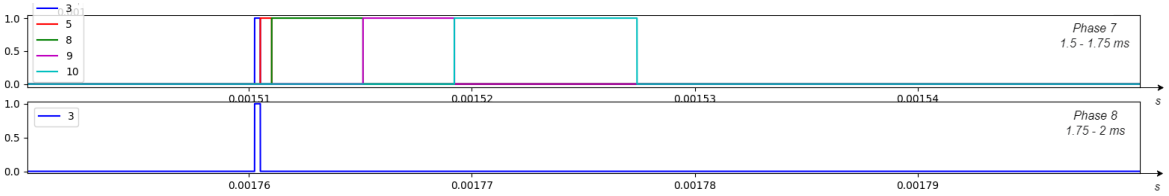


Figura 5.11: Configuración de 20 flujos en un enlace (ampliado)

Por tanto, en este enlace se ha logrado llevar a cabo una planificación en la que no aparecen huecos mas que el caso especial comentado, con lo que la cola de *best-effort* podrá ser transmitida ininterrumpidamente desde el valor de UP hasta el LW de la siguiente fase. En cambio, si nos vamos al enlace existente entre los nodos 0 y 2 ahora sí nos encontramos con la aparición de un hueco entre los tiempos del final de la transmisión del flujo 0 y del inicio del flujo 18, tal y como se muestra en la Figura 5.12. Esto se debe al retraso del flujo 18 en el siguiente salto tal y como se ha visto en el Apartado 5.1.5, donde se retrasan los tiempos de transmisión designados a éste con el fin de adecuar su tiempo de llegada al final de la transmisión del flujo 17 para minimizar el retardo introducido por la espera en cola a costa de introducir tal hueco. De hecho, si comprobamos su retardo extremo a extremo, podemos observar que su valor coincide con el de otros flujos de las mismas características como el flujo 13, el cual no ha sido retrasado y también se compone de tramas de 256 Bytes y con dos saltos pero con una ruta diferente. De este modo, el tiempo transcurrido desde la llegada de la trama al nodo hasta el comienzo de su transmisión será cero. Con ello, el intervalo de tiempo desplazado así como el asignado al hueco es del valor de *dif*, es decir, la diferencia entre UP y el instante de llegada inicial t_0.

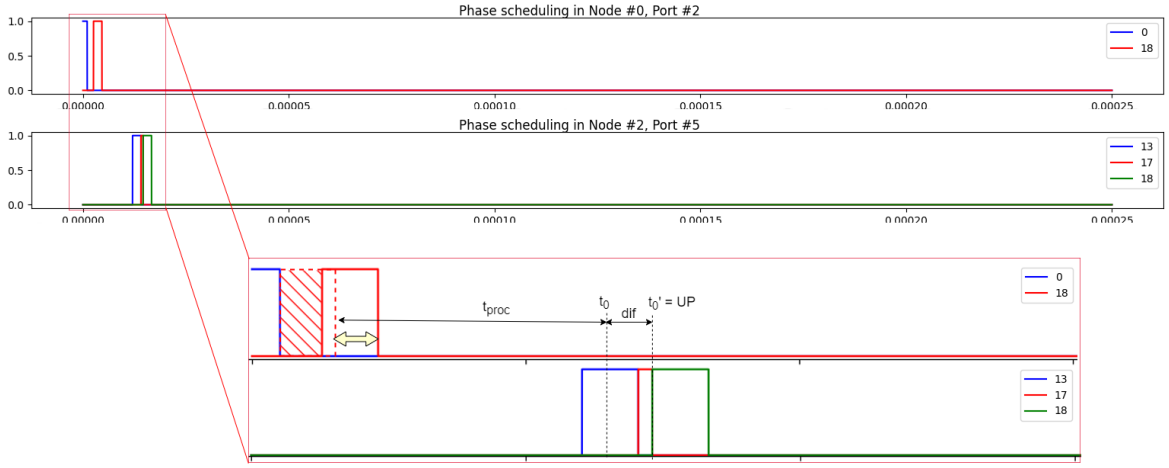


Figura 5.12: Retraso en la planificación del flujo 18 con *zero-wait*

Visiblemente, dicho hueco no podrá ser utilizado por otras colas dado que no hay espacio suficiente para albergar ninguna trama ni su correspondiente banda de guarda para evitar colisiones, de modo que se penalizará esta solución. Estos huecos se miden a través de los intervalos restantes después de tratar de repartir el espacio para tramas de 1542 Bytes con la MTU en Ethernet, pues recordemos que se penaliza en base al peor de los casos en el que tampoco se produce transmisión alguna durante la banda de guarda. Además, con esto se penaliza también la cantidad de veces con la que aparecen estos huecos dado que implica cierto retardo en el que el GCL conmuta a otra cola en un pequeño lapso de tiempo en el que tampoco se está transmitiendo nada y que no se están considerando en este modelo ya que intervienen durante las bandas de guarda. De tratarse de una cola reservada a otro tipo de tráfico de prioridad con tamaños de trama fijos, este criterio cobra aún más sentido ya que en este caso no sería posible insertar tramas en los huecos de ninguna forma, con lo que el hueco no podría ser utilizado. De este modo, el criterio de penalización adoptado ha de ser en todo caso la MTU más pequeña de entre todas las colas restantes, tratándose así del caso *best-effort*.

Con ello, se ha obtenido un retardo medio de 34.2 μs entre todos los flujos y con una utilización del 99.36 % de los enlaces. Sin embargo, la evaluación del retardo no se hace a través de estos valores, sino que se hace con el valor normalizado del retardo de todos y cada uno de estos flujos entre su retardo máximo tolerable, lo que penalizará más a aquellos que se acerquen a dicho límite y favorecerá aquellas soluciones en las que se minimice dicho retardo para flujos con menor retardo límite. Este aspecto lo estudiaremos con más detalle con la función *fitness* del Apartado 5.2.2. Esta comprobación del funcionamiento del planificador TSN síncrono se ha tratado de un caso más simple, en el que tan solo se ha trabajado con 20 flujos. Sin embargo, en el siguiente Apartado 5.2 nos encontraremos con casos peores con un mayor número de flujos y características diferentes con el fin de evaluar el rendimiento de este sistema planificador basado en algoritmos genéticos. Con ello, a los flujos de la Tabla 5.2 se le irán añadiendo nuevos con características diferentes.

5.2. Implementación del Algoritmo Genético

Para el desarrollo del algoritmo genético encargado de optimizar la solución del sistema planificador visto hasta ahora a través de la combinatoria se ha hecho uso de la librería PyGAD, la cuál ya se ha descrito en el Apartado 4.2.2. Con ello, se procederá a analizar los diferentes elementos que componen al algoritmo genético para obtener un mayor rendimiento tanto en la ejecución como en la evaluación del modelo.

5.2.1. Población, cromosomas y genes

Para comenzar a utilizar PyGAD, es imprescindible conocer antes el mecanismo poblacional con el que se pretende codificar y hacer funcionar el algoritmo genético. Esta librería acepta diversos formatos para los genes que conforman el cromosoma, como ya se vio con los parámetros *gene_type* y *gene_space*. Recordemos que cada uno de los genes se asocia con la ruta seguida por un flujo determinado, con lo que la dimensión del cromosoma coincide con el número de flujos M. Sin embargo, dado que se trabaja con un espacio de búsqueda basado en un vector de posibles rutas restringido (Algoritmo 1), resulta interesante hacer uso de un identificador de número entero positivo que señalice a una ruta de entre las establecidas. Este identificador corresponde por tanto con la posición

de la ruta en dicho vector. De esta forma, las posibilidades de selección de una ruta u otra por el algoritmo genético quedarán determinadas por la dimensión de estos vectores de rutas. Por ejemplo, en la Figura 5.13 se presenta el caso en el que el espacio de búsqueda se ha restringido a únicamente seis rutas posibles para todos los flujos, pese a que existen otras alternativas más para algunos.

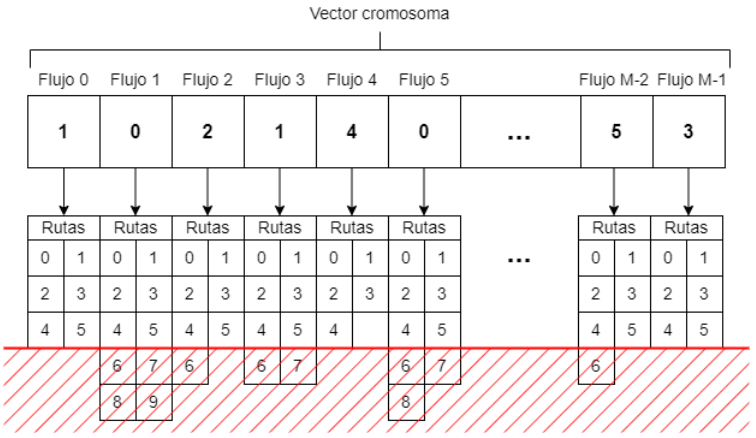


Figura 5.13: Vector de rutas de flujos en cromosoma con espacio de búsqueda restringido

Inicialización de la población

Con ello, para una solución determinada cada gen adquirirá uno de estos valores, es decir, una ruta. De este modo, se puede generar una población inicial de soluciones en las que cada gen adquiere un valor de carácter aleatorio dentro del espacio de búsqueda a través del parámetro *initial_population*. En función de la complejidad, y siempre que exista mutación, se puede optar bien por ampliar este tamaño de población, la cual se mantendrá a lo largo de las generaciones; o bien por elevar el número de generaciones límite. Por ejemplo, en este caso se empleará una población de 100 soluciones con un límite de 15.000 generaciones, lo que se podría limitar a un criterio de parada por saturación de 5.000 generaciones. Por tanto, puesto que se va a ejecutar el modelo de planificación para cada individuo, es muy importante restablecer antes los valores de tiempos relativos y globales reservados a cada flujo en todos los nodos TSN de la red.

También cabe la posibilidad de sustituir una de estas soluciones de genes aleatorios por una solución propia a esta población inicial de modo que parta de un resultado apropiado como es el caso de seleccionar uno de los caminos más cortos para todos los flujos e impulse el resultado con un mayor valor de *fitness*. De esta forma, con la mutación de los genes se podrá lograr una convergencia más rápida en los casos donde el espacio de búsqueda sea demasiado amplio para un determinado número de generaciones, aunque esto podría sesgar en cierto modo dicho espacio y alejarse de la solución óptima. Sin embargo, se estaría limitando la convergencia hacia escasas trayectorias.

5.2.2. Evaluación de la solución. Función *fitness*

La función *fitness* es la encargada de evaluar una a una las diferentes soluciones que conforman una generación en la población con el fin de poder ir mejorando los resultados a medida que avanza su ejecución. Por tanto, esta función de evaluación tiene una gran

relevancia en este proyecto ya que a partir de ella se pueden llegar a optimizar los resultados. De este modo, dado que se pretende considerar tanto el retardo extremo a extremo de cada uno de los flujos como la utilización de los enlaces, se ha realizado el cálculo de los mismos para su evaluación.

Para medir el retardo extremo a extremo de un flujo, tal y como se presentó en la Ecuación 5.2, se ha medido primero la diferencia de tiempos entre el inicio de la transmisión en el primer nodo con el final de la misma en el último, tratándose así del tiempo en el que la trama se ha estado moviendo dentro de la red TSN. Posteriormente, se le han añadido los tiempos de procesamiento característicos de los nodos TSN limítrofes en su llegada a la red así como en la salida para luego añadirle también el retardo que se produce en la transmisión de la trama a través de los enlaces que conectan directamente a los dispositivos transmisor y receptor. Nótese que estos tiempos de procesamiento son nulos en la planificación ya que se considera un instante t_0 igual a cero en su planificación en el primer salto y no se planifica en el último. Una vez calculado este retardo, se normaliza en función del retardo máximo tolerado y se promedia con el resto de flujos.

Por otro lado, se miden los intervalos correspondientes a todos los huecos producidos en las múltiples fases de los enlaces que conectan a los distintos nodos y se comprueba uno a uno si se puede llegar a introducir una trama MTU Ethernet. Al computarse para todas las fases, si se tratara de un esquema *half-duplex* este tamaño total se ha de reducir a la mitad ya que las fases operarían sobre un mismo enlace en ambos sentidos. En el caso de que esto no sea posible, el tiempo añadido al total como no aprovechable es el propio tamaño del hueco, mientras que si en él caben al menos dos tramas (una de información y otra como banda de guarda) se computará únicamente la banda de guarda.

Ambos valores, tanto retardos como huecos, se sumarán con el fin de proporcionar el valor *fitness*. Dado que PyGAD maximiza este valor y en cambio un mayor retardo así como un mayor tamaño total de huecos penalizan, se ha de invertir la suma de ambos, tal y como se ve en la Ecuación 5.3. Sin embargo, puesto que la magnitud de los retardos y la de los huecos no es la misma, se ha de utilizar un factor corrector (τ_{corr}) para que pueda nivelar el caso en el que se pretenda evaluar ambos por igual. Dicho valor se ha estimado en torno a 5, es decir, el retardo es aproximadamente igual a cinco veces el de la suma de los huecos. Esto puede variar considerando que, en función de los tamaños y los nodos origen y destino de cada flujo del conjunto utilizado, el retardo normalizado de los mismos se verá también afectado. Además, a medida que incremente el número de flujos en la planificación es muy probable que se eleve consigo el número de huecos. Sin embargo, basta con equilibrar tan solo dicha magnitud para evitar que una pequeña mejora de uno de los parámetros sature un considerable empeoramiento de la otra.

$$fitness = \frac{1}{weight_{delay} \cdot \frac{normDelay_{total}}{numFlows} + weight_{gap} \cdot \frac{gaps_{total}}{\#phases \cdot \#ports} \cdot \tau_{corr}} \tag{5.3}$$

Adicionalmente, es posible centrarnos en un único parámetro a través de los pesos, de modo que se le dé especial relevancia al de mayor valor. Con esto, el resultado del *fitness* empeorará ya que se está dividiendo por un valor aún mayor después de realizar el producto por estos pesos, pero esto no es relevante ya que la escala de esta función no tiene mayor importancia que la de poder consultar la mejora del algoritmo genético, pues lo verdaderamente importante son los retardos y la utilización de los enlaces, como veremos más adelante. De esta forma, no se tendrá en cuenta este valor *fitness* en la comparación de los resultados de ejecuciones con diferentes volúmenes de flujos.

Una vez desarrollados todos estos aspectos relacionados con la implementación del modelo y de su evaluación a través de la función *fitness* del algoritmo genético, podemos pasar a comprobar primero la mejor combinación de métodos de selección y cruce de forma que las soluciones puedan converger mejor hacia la óptima. Asimismo, se pueden probar distintas probabilidades de mutación.

5.2.3. Selección y emparejamiento de los individuos

Para ello, se ha realizado una batería de cinco ejecuciones para cada combinación de los métodos ofrecidos por la librería PyGAD a través de los parámetros *parent_selection_type* y *crossover_type*; y se ha calculado y representado el valor promedio de cada uno de ellos ya que una ejecución puede puede variar respecto a otra debido a la aleatoriedad de la población inicial elegida. En este caso se cuenta con los métodos de selección aleatoria, basada en ranking de individuos a través del *fitness*, por ruleta y por torneos; además de los métodos de cruce en un único punto, en dos puntos y de elección aleatoria de los genes de cada padre. Con esto, se han empleado los mismos cinco conjuntos de soluciones iniciales con el fin de evitar variabilidad entre las doce combinaciones posibles. En este caso se ha elevado el número de flujos hasta treinta con el fin de comprobar los resultados con una mayor variedad de éstos. Cabe a destacar que se ha optado por la opción de elitismo, de modo que el resultado de cada generación siempre iguale o mejore la anterior. Además, se ha configurado inicialmente una probabilidad de mutación del 10 % sobre cada gen, aunque en el Apartado 5.2.4 se comprobará el comportamiento de este parámetro con más detenimiento. El resultado de ello es el mostrado en la Figura 5.14.

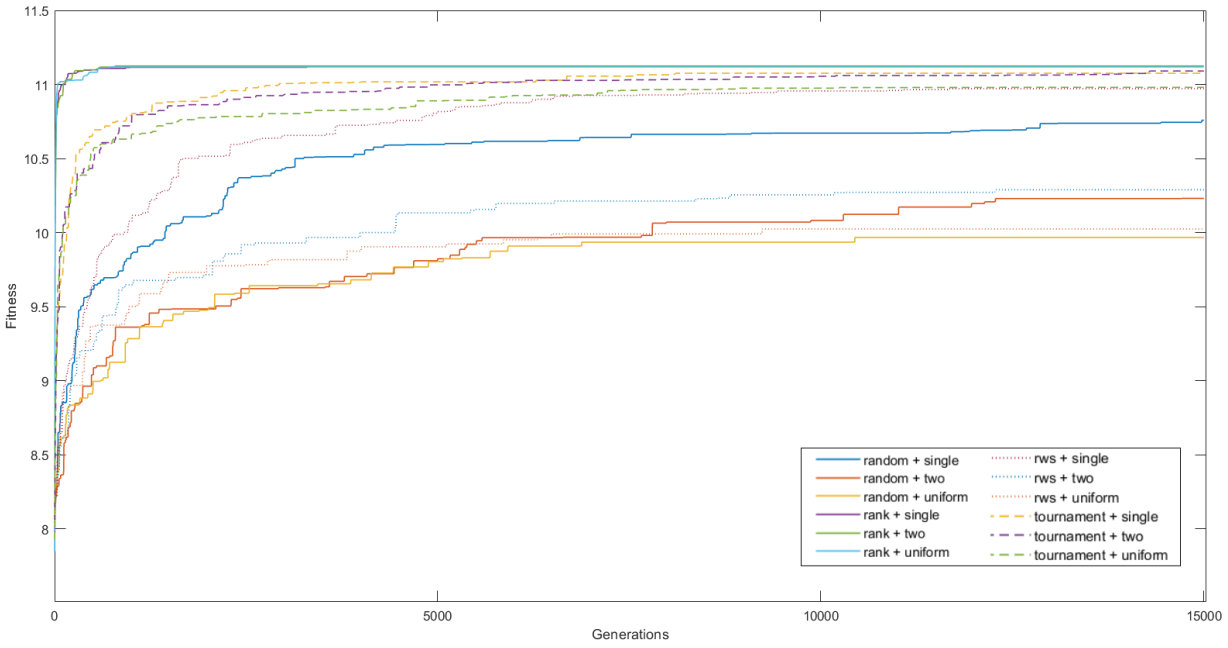


Figura 5.14: Comparación entre combinaciones de métodos de selección y cruce en PyGAD

Se tiene por tanto que la selección por ranking parece mejorar considerablemente los resultados con una convergencia más rápida hacia lo que apunta ser la solución óptima, pues tan solo han sido necesarias algo menos de 1.500 generaciones. Otras combinaciones como la basada en torneos no se quedan atrás, principalmente las de cruce en un único

punto y en dos puntos del cromosoma, aunque por lo general se han visto soluciones algo más alejadas del objetivo y con un mayor número de generaciones necesario para llegar a ello. Una clara alternativa a los torneos puede ser el uso de ruletas ya que mantiene una alta presión selectiva, siempre y cuando se produzca también en un único punto, pues de lo contrario esto empeorará los resultados al igual que ocurre con la selección aleatoria. Sin embargo, parece que con este último método de selección es posible acercarse al resultado logrado con los torneos en el caso de también realizar el cruce en un único punto ya que ofrece intermedios. De todas estas últimas combinaciones, principalmente con el caso aleatorio, puede apreciarse el estancamiento del *fitness* durante varias generaciones ya que, bien por la presión selectiva o bien por la aleatoriedad, no se logra combinar adecuadamente las soluciones para mejorar el resultado y con ello el padre elitista perdura durante un gran número de ellas. Por lo general, se ha visto también que el cruce en un único punto parece mejorar bastante en todos los casos, salvo el de ranking donde los resultados son muy similares, aunque el caso de cruce en dos puntos parece reducir el número de generaciones necesario y se ha acercado más a la mejor solución encontrada. Por el contrario, el cruce de genes aleatorio uniforme no parece ser conveniente en ninguno de ellos ya que de nuevo, para todos los métodos de selección salvo para el de ranking, empeora el resultado. Esto puede deberse principalmente a que, por el hecho de seleccionar ambos puntos del cromosoma de forma aleatoria, tenga una mayor flexibilidad para modificar únicamente un pequeño subconjunto de genes intermedio de modo que uno de los descendentes en la nueva generación haya adquirido la mejor solución como padre y además se hayan modificado únicamente los genes adecuados. En cambio, también se ha comprobado este mismo procedimiento para un mayor número de flujos tal cual puede verse con la Figura 5.15, donde esta vez se ha ejecutado el sistema de planificación para un total de 80 flujos.

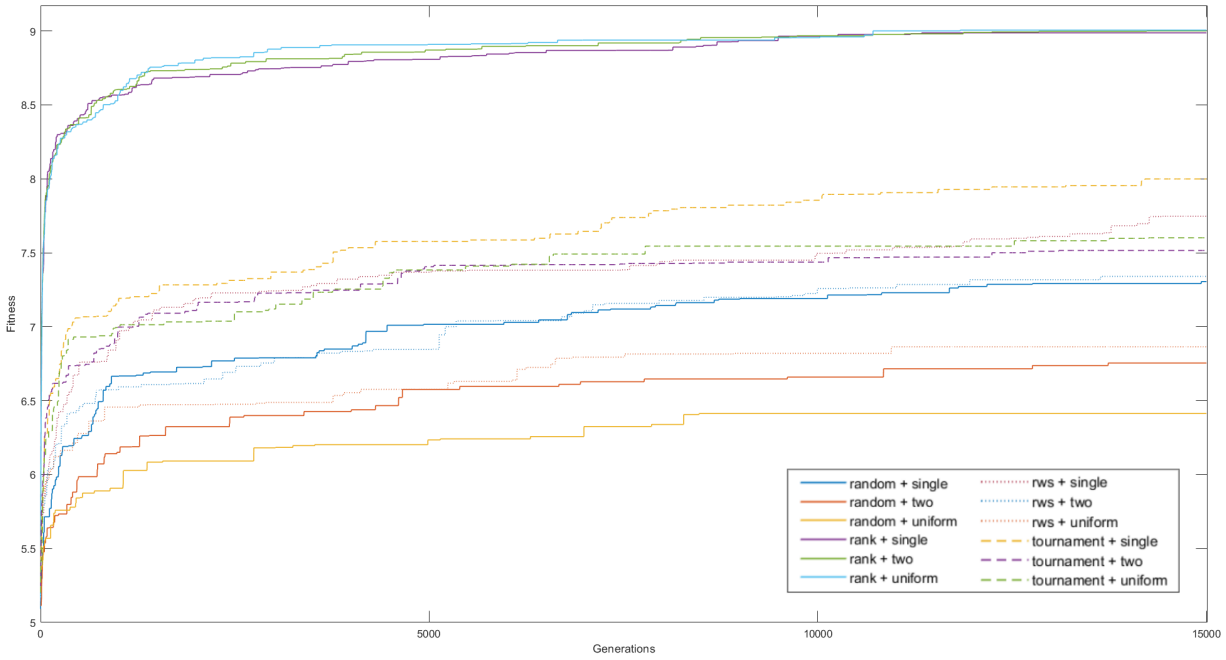


Figura 5.15: Comparación entre métodos de selección y cruce para 80 flujos

Ahora, la diferencia entre el hecho de utilizar un método de selección u otro es aún más clara ya que la alternativa que más se aproximaba a los mejores resultados logrados por la selección basada en ranking, la de torneos, se ha descolgado bastante debido a que

requiere de un mayor número de generaciones sin llegar a acercarse a los mejores resultados. Ocurre algo similar para el resto de métodos de selección, aunque la selección basada en ruleta y con un único punto de cruce parece incluso mejorar ahora los resultados de los torneos. De hecho, la tendencia es continuamente creciente pese a verse por el límite en el número máximo de generaciones marcado en 15.000. De este modo, la función podría seguir escalando hasta conseguir un mejor resultado, aunque esto significaría elevar el tiempo de cómputo para ni siquiera mejorarlo. Asimismo, la selección por ranking también se ha visto afectada dado que también ha necesitado un mayor número de generaciones para dar con la solución de mayor *fitness*, aunque en todos los casos parece estabilizarse en torno a las 12.000 generaciones. Nótese además que el valor de máximo *fitness* logrado queda ahora por debajo del caso de tan solo 30 flujos ya que ha aparecido un mayor número de huecos que han penalizado más a estas soluciones. Además, un mayor número de genes aumenta el espacio de búsqueda, dificultando la convergencia y estancando aún más la función *fitness*.

Por tanto, se puede confirmar la gran ventaja que posee el uso del método de ranking de individuos sobre el resto. Sin embargo, la elección del método de cruce dentro de la combinación con el método de selección de por ranking continua siendo bastante difusa ya que logra resultados y tiempos de convergencia muy similares entre sí. De este modo, resulta coherente decantarnos por dicho método de selección junto con el método de cruce en dos puntos del cromosoma, aunque los otros métodos cruce pueden valer del mismo modo. Una vez conocido esto, se puede pasar a estudiar el comportamiento de esta combinación con diferentes probabilidades de mutación con el fin de seleccionar aquella que más nos convenga.

5.2.4. Mutación de los genes del cromosoma

Para la mutación de genes dentro de cada solución, se ha procedido del mismo modo que con las combinaciones entre métodos de selección y cruce, es decir, llevando a cabo el promedio de cinco ejecuciones para diferentes probabilidades de mutación con el parámetro *mutation_probability*. Con esto, se han probado un total de diez casos con probabilidades desde el 0 % hasta el 50 %. Recordemos que esta probabilidad se aplica a cada gen, con lo que el número de mutaciones por cromosoma puede oscilar en función de esta probabilidad, pudiendo tratarse así de un mayor número de genes a medida que ésta se incrementa. El resultado de estas pruebas puede verse en la Figura 5.16. Como parece lógico, de no emplearse mutación, la selección quedará saturada en una misma solución al cabo de unas pocas generaciones de modo que no podrá modificar sus genes y se mantendrá con el mismo resultado hasta el final de la ejecución. En cambio, con el uso de probabilidades en torno a un 5-25 % se elevan las posibilidades de lograr una buena solución y en un número de generaciones más razonable. Sin embargo, a partir del 30 % se comprueba que el hecho de aumentar esta probabilidad de mutación de los genes no implica la mejora del resultado o del número de generaciones necesarias para la convergencia, pues va empeorando el resultado a medida que ésta se aproxima al 50 %. En este caso se aprecia además que la función *fitness* comienza a quedarse estancada en torno a las 5.000 generaciones dado que no es capaz de mejorar el resultado con tantas mutaciones en un mismo cromosoma, pues si la mutación de un gen concreto podría mejorarlo no lo conseguirá si el resto se ven también modificados. Si se pretende mejorar por tanto dichos resultados evitando que la función *fitness* quede estancada en una misma solución será necesario aumentar aún más los tamaños de la población.

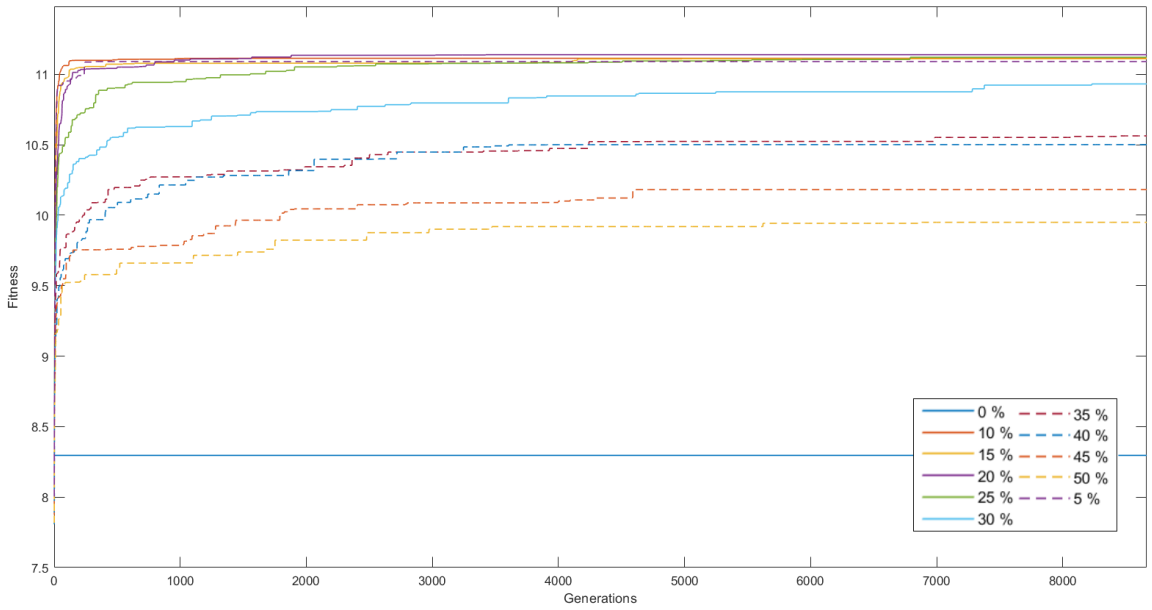


Figura 5.16: Comparación entre diferentes probabilidades con PyGAD

De este modo, parece razonable continuar empleando una probabilidad de mutación relativamente baja, pues la diferencia dentro de este rango parece difusa pues también parece que un 5 % logra mejorar ligeramente el resultado en cuanto al número de veces que la ha mejorado, aunque recordemos que se han computado los valores promedio. Es por ello por lo que estos resultados son más bien orientativos para decantarnos por una combinación u otra. Sin embargo, a medida que se aumenta el número de flujos y con ello el espacio de búsqueda una menor probabilidad podría llevar menos tiempo hasta la convergencia. Con esto, se ha procedido a comprobar esto mismo con el caso anterior de los 80 flujos y que se puede ver en la Figura 5.17. Esto puede deberse principalmente a que el hecho de modificar tan solo las rutas correspondientes a un minúsculo subconjunto de flujos puede contribuir notablemente a la mejora en lugar de si lo hacen varios a la vez, lo que demuestra el buen rendimiento que puede ofrecernos un algoritmo genético pese a que no siempre logre alcanzar la mejor solución en un número limitado de generaciones. De este modo, estas pequeñas variaciones pueden también provocar la convergencia hacia máximos locales en lugar de máximos globales dado que tan solo se realizan pequeños cambios sobre los cromosomas de los individuos de forma que una vez se han llevado a cabo un mayor número de generaciones éstos son realmente similares entre sí (si no iguales) debido al proceso de selección de los mejores individuos tras el emparejamiento, por lo que si el genoma del máximo global distara en cuanto a número de genes diferentes de la solución actual sería difícil llegar hasta él aunque, como se puede ver en los resultados de esta prueba, no resulta conveniente aumentar esta probabilidad de mutación por encima del 10 % ya que empeora el rendimiento con un menor valor *fitness* en el máximo fijado a 15.000 generaciones. Por tanto, se ha concluido con que una combinación del método de selección basado en ranking junto con el cruce en dos puntos y una mutación del 5 % puede ser suficiente para probar el modelo con un algoritmo genético dado el mayor valor *fitness* dentro de los límites establecidos así como por su rápida convergencia; aunque también es cierto que la curva de la probabilidad de mutación del 10 % es creciente hasta las últimas generaciones si llegar a mejorar a otras probabilidades menores antes de esas 15.000 generaciones. Sin embargo, es muy importante tener en cuenta las limitaciones de los

algoritmos genéticos ya que la heurística en la que se basan provoca pequeñas variaciones en cuanto a la convergencia hacia un resultado u otro con valores *fitness* cercanos, pues se basa en poblaciones iniciales, emparejamientos y cruzamientos aleatorios. Por tanto, para evitar en mayor medida esta variabilidad, se ha decidido descartar algunos parámetros como la probabilidad de selección así como la de cruce de los que PyGAD dispone.

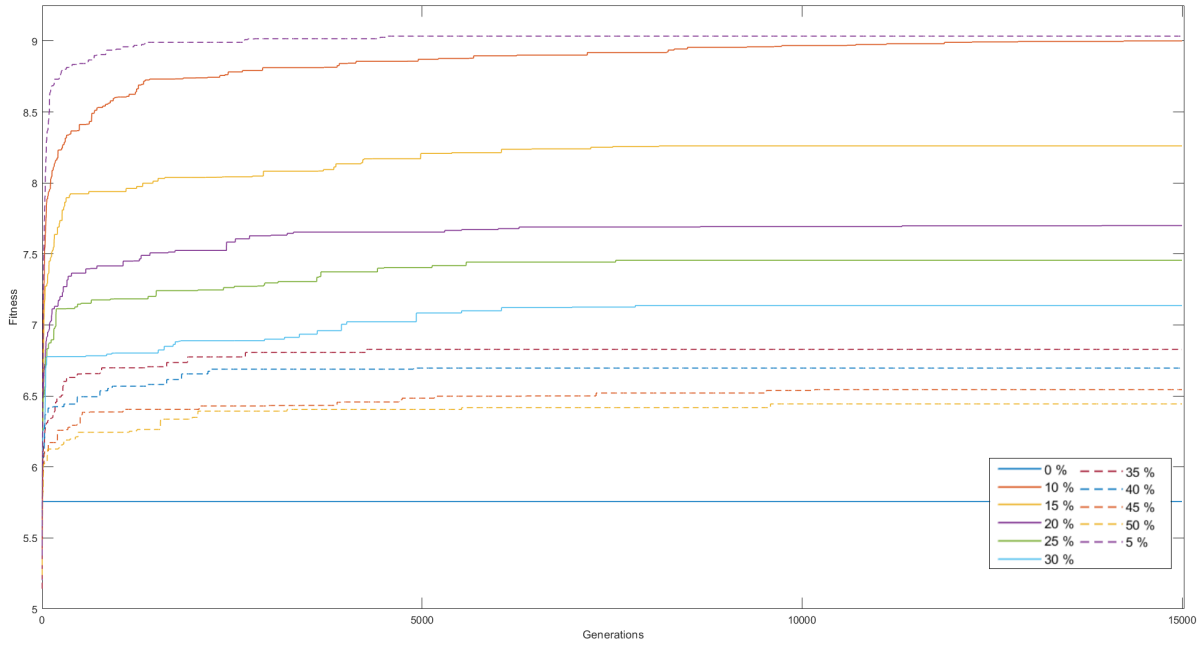


Figura 5.17: Comparación entre diferentes probabilidades para 80 flujos

5.2.5. Resultados del algoritmo genético sobre el modelo

Una vez constituido el modelo de planificador TAS y comprobado su rendimiento a través de las diferentes combinaciones de métodos del algoritmo genético para su optimización, el sistema ya está listo para ser analizado a través de los pesos del retardo y la utilización del enlace establecidos en la función *fitness*, su comportamiento en función del número de flujos y la respuesta ante fallos a través de *Machine Learning*.

Ajuste de los pesos del retardo y la utilización del enlace

Como se ha visto con la Ecuación 5.3, es posible ajustar los pesos de ambos parámetros con el fin de poder optimizar exclusivamente uno de ellos a costa de empeorar el otro. Esto puede resultar muy interesante en función de los requisitos de los flujos así como del margen de éstos respecto a su retardo máximo tolerado, pues si éste es lo suficientemente amplio se puede jugar aumentando el peso de la evaluación de los huecos para reducirlos lo máximo posible dentro de los límites a costa de aumentar ligeramente dicho retardo. En la Figura 5.18 se muestra un barrido de estos pesos para la planificación de 80 flujos, en el que se comprueba el resultado del retardo promedio de los flujos frente al de utilización. De este modo, se pesan ambos parámetros con valores entre 0 y 1 y cuya suma entre ellos sea siempre la unidad. Para ser más representativo, se ha medido el tiempo porcentual de las fases sin utilizar en la planificación debido a los huecos que se producen respecto al periodo mínimo así como las bandas de guarda adicionales empleadas. En este caso se representa

la ejecución que mejor resultado de *fitness* ha ofrecido para cada ratio de un total de cinco para ser más justos frente a la aleatoriedad de la población inicial pues, como se ha visto, en algunos casos la mejor solución encontrada podía variar dado el límite de 5.000 generaciones sin cambios para limitar el tiempo de cómputo.

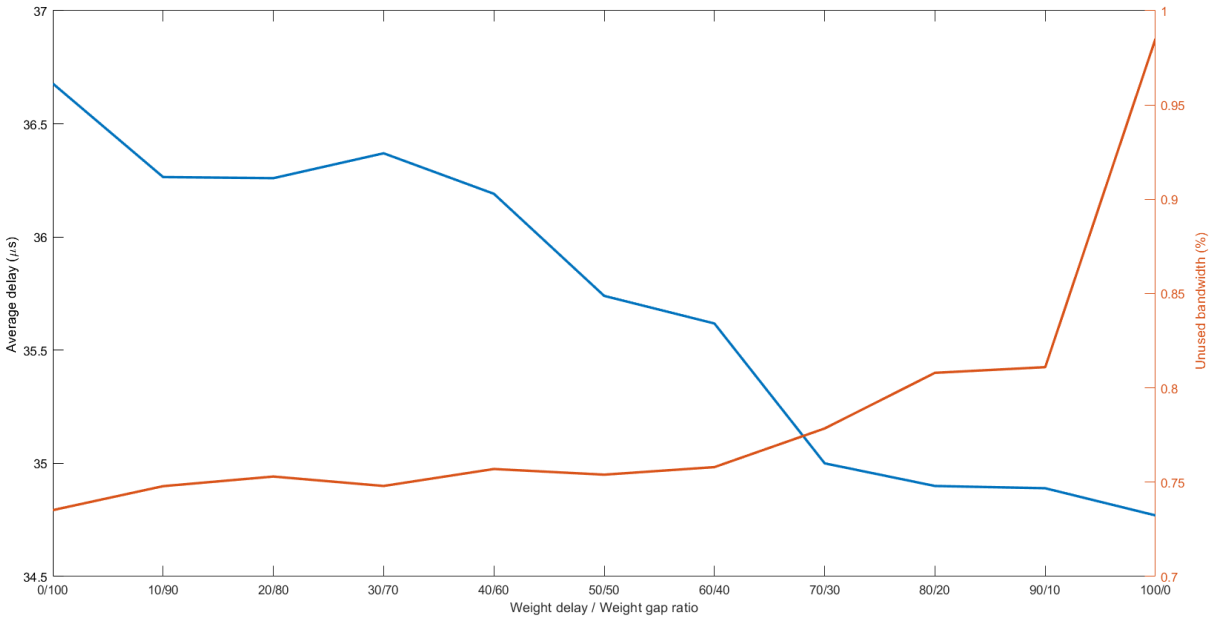


Figura 5.18: Comparación entre diferentes pesos en la función *fitness*

Como se puede ver, existe una clara tendencia de la función de pesos en la que, a medida que el peso del retardo aumenta, su valor promedio logra verse generalmente reducido, aunque en cambio se produce un aumento del tiempo no utilizado en la planificación debido a la aparición de estos huecos y de las bandas de guarda. Por el contrario, reducir dicho ratio entre retardo y huecos implicará el aumento del retardo promedio para aumentar la utilización de los enlaces, con lo que se tiene una función de los mismos que depende de estos pesos y que puede servir para optimizar la planificación de la red TSN en función de las necesidades. Se ha de tener en cuenta que actualmente se trabaja sobre un espacio de búsqueda acotado y por ende sin todas las posibles rutas para cada flujo, por lo que otras rutas alternativas no contempladas en el mismo podrían permitir una mayor mejora en cuanto al resultado de los huecos y evitar así las variaciones que se observan en la gráfica en torno a 30/70 (relación sobre 100), donde se producen ligeros aumentos del retardo extremo a extremo conforme los huecos pierden peso con el fin de ajustarse a éstos últimos. En dicho punto se produce por tanto también una ligera reducción del tiempo en desuso de modo que en este caso no se ha encontrado una solución que optimizara algo más el retardo que los puntos anteriores y que por tanto maximice del todo el *fitness*, aunque se ha conseguido jugar con la importancia de cada parámetro a través de dichos pesos. Sin embargo, este retardo no puede verse más reducido que el caso en el que todos los flujos sigan la ruta con menor número de saltos al emplearse este esquema de *zero-wait*, por lo que es probable que a un determinado ratio próximo a 100/0 se logre dicho valor con el que se llegue a estancar el resultado de este parámetro y, por tanto, la evaluación de los huecos pueda seguir empeorando a medida que se le resta importancia, sobre todo si en el caso de excluirlos de la evaluación con un peso nulo como ocurre con ese 100/0. En este caso, nos encontramos con una diferencia en el retardo de aproximadamente 2 μs frente a

una pérdida del 0.25 % del tiempo planificado entre los casos de excluir un parámetro u otro con pesos de nulidad. Se ha observado que en el caso de anular uno de los parámetros como pueden ser los huecos su valor puede dispararse hacia un rango más amplio dado que no se está teniendo en cuenta para la evaluación de cada individuo, por lo que una solución con el mismo *fitness* puede tener igual retardo pero utilización de enlaces completamente diferentes, por lo que se desaconseja el uso de estos valores nulos para los pesos, pues es necesario que, por mínimo que sea, se tenga en cuenta en la función *fitness*. Además, de aumentarse el número de flujos o desplegarse una topología de red TSN más compleja, podrán verse resultados aún más distinguidos.

Planificación en función del número de flujos

Este retardo extremo a extremo promediado entre todos los flujos así como el nivel de utilización de los enlaces han de entenderse únicamente en relación al volumen de flujos planificados y sus características pues, al haberse aleatorizado los parámetros que contribuyen al orden de los mismos (e.g. tamaño de las tramas, origen y destino y con ello el mínimo número de saltos, etc.), estos resultados pueden verse alterados en función de la cohesión que pueda existir entre ellos, aunque puede apreciarse con ello una clara tendencia al aumento en cuanto al retardo promedio en la red TSN así como al tiempo no planificado para ningún tipo de tráfico, tal y como se muestra en la Figura 5.19, donde se ejecutan todos ellos con un ratio unitario 50/50 en el que se igualan ambos pesos.

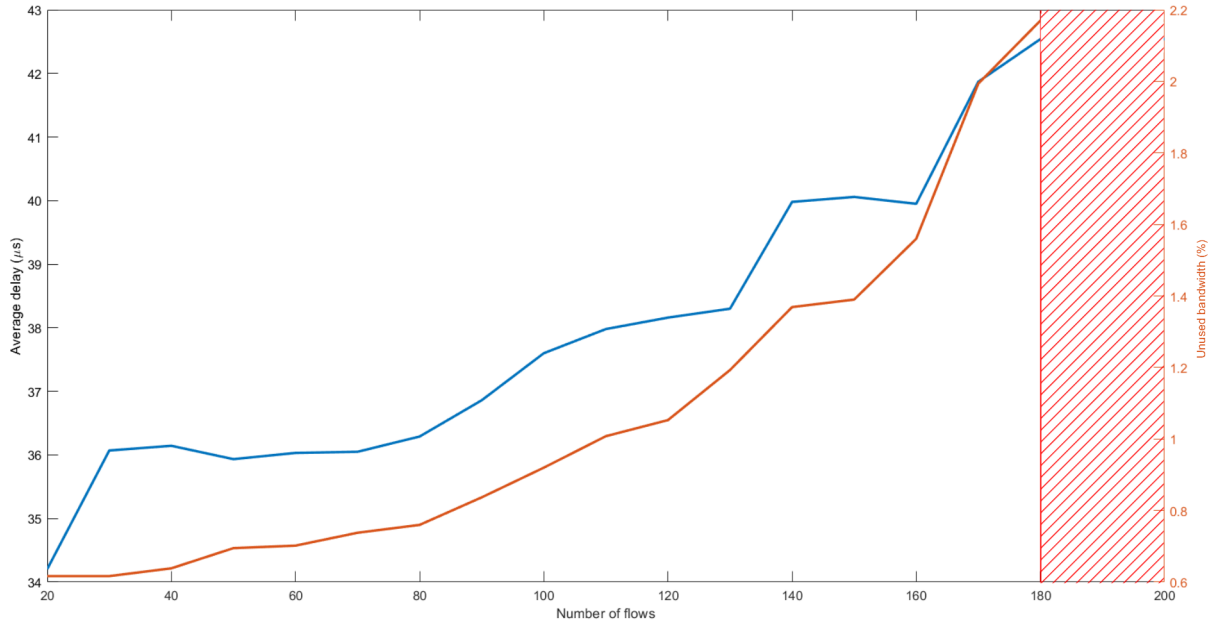


Figura 5.19: Comparación en función del número de flujos

Esto se produce dado que la evaluación de huecos empeora a medida que se planifican un mayor número de flujos, con lo que el retardo lo hace también consigo para tratar de equilibrar el resultado y mantener en medida de lo posible el *fitness*, por lo que se obtienen peores resultados a medida que se incluyen nuevos flujos respecto a la Tabla 5.2 y muestra la necesidad de ajustar el factor de corrección. Además, se producen dos importantes saltos en lo que a retardos extremo a extremo promedio y huecos se refiere:

el primero a partir de los 30 flujos y el segundo una vez se alcanzan los 120 flujos; lo cual puede deberse principalmente a esta casuística comentada, pues también se observa una ligera caída del mismo entre los 40 y los 50 flujos. En el caso del nivel porcentual de utilización de los enlace se puede ver que cae de forma continua, aumentando así el tiempo en desuso desde un 0.6 % hasta cerca del 2 %. Sin embargo, llega a un punto a partir de los 190 flujos en el que no logra encontrar una solución posible dentro del límite de las 5.000 primeras generaciones, es decir, con un valor *fitness* nulo; por lo que se tendrá bien que aumentar dicho límite por saturación de la función *fitness* que permita encontrar nuevas combinaciones que así lo permitan o bien aumentar el margen de fase visto con la Figura 5.8 (e.g. 20 %) dado que es muy posible que exista un flujo que precise de más de esos 25 μs dado que requiere un mayor número de saltos en la red TSN para llegar a su destino y dentro de la misma fase, algo que ocurre con los últimos cuando, pese a probar diferentes rutas, no se encuentra una que permita su planificación en un nodo intermedio debido al tiempo restante hasta el fin de la fase, como ya se ha comentado con anterioridad.

Esto también puede realizarse de nuevo para los casos con menor y mayor ratio a través de los pesos, así pudiendo establecer los límites entre los que se mueven ambos parámetros a través de los diferentes ratios 90/10 y 10/90 para cada volumen de flujos. En la Figura 5.20 pueden apreciarse dichos límites de retardo superior e inferior para el caso de los huecos, a partir de los cuales no se consiguen valores más grandes (azul) o más pequeños (naranja), respectivamente. Nótese que ahora estos límites siguen oscilando dada la aleatoriedad sobre las características de los flujos y el espacio de búsqueda, pero se mantienen algo más continuos para contener dentro de ellos la curva creciente vista con la Figura 5.19, también representada en esta gráfica en morado.

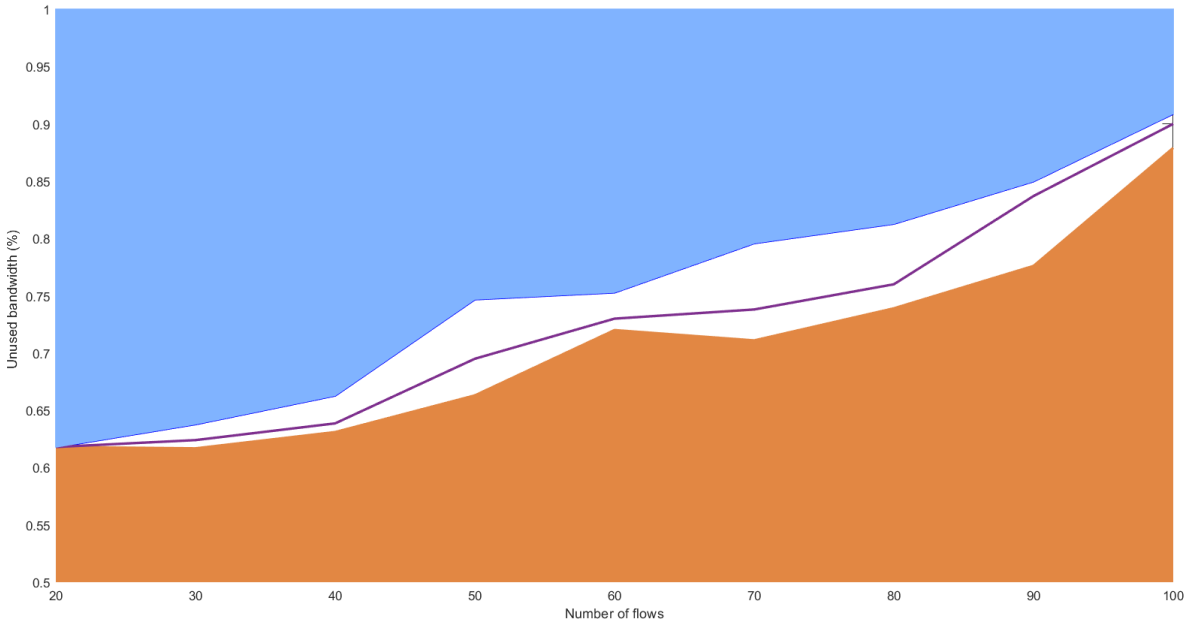


Figura 5.20: Límites del peso de los huecos en función del número de flujos

En este caso el valor de los huecos aumenta irremediablemente conforme se planifican nuevos flujos con este esquema *zero-wait*. Resulta lógico pensar que en ambos casos todos los valores permanecen dentro de la zona blanca delimitada por los valores que establecen ambos ratios, especialmente la curva con el ratio 50/50 que queda aproximadamente a la mitad entre ambos límites ya que se trata de optimizar ambos parámetros por igual.

Capacidad de respuesta ante posibles fallos en la red TSN

El sistema ha de ser entrenado a través del algoritmo genético de forma que, de producirse cualquier fallo en un enlace, sea lo suficientemente robusto frente a éstos al encargarse el controlador de recuperar las rutas para cada flujo de modo que tan solo se vea afectado en el menor tiempo posible desde su detección. Por ello es necesario ejecutar previamente al despliegue dicho algoritmo para todo posible escenario a través de una serie de cambios sobre la topología de la red TSN, llegando incluso a contemplar la caída de dos o más nodos de forma simultánea. De este modo ya no será necesaria ninguna heurística adicional que se encargue de recalcular en tiempo real el camino más adecuado para cada flujo o incluso que se realice de forma que pueda perjudicar los resultados conseguidos hasta ahora como se ha presentado en alguno de los trabajos tratados en el estado del arte de este proyecto. Si, por ejemplo, volvemos al caso más sencillo de 20 flujos presentado en la Tabla 5.2 y desconectamos los enlaces que comunican al nodo 0 con el nodo 2 y viceversa (véase Figura 5.2), veremos que ahora el tráfico correspondiente a los flujos #0 y #18 se han visto reubicados en el único enlace que conecta al nodo 0 con el resto de la red, es decir, el que comunica con el nodo 1. Del mismo modo, el flujo #12 también se ha visto desviado en el enlace correspondiente al otro sentido, es decir, el que une al nodo 1 con el nodo 0. No obstante, el hecho de haber tratado de optimizarlo mediante el algoritmo genético resulta también en cambios adicionales en las rutas. De esta forma, la nueva Tabla 5.3 refleja la situación actual.

ID	nodo *src*	nodo *dst*	T_{tx}	$Delay_{max}$	*tamaño*	Ruta	$Delay_{e2e}$
#0	0	3	250 μs	250 μs	128 Bytes	0-1-3	34.096 μs
#1	0	6	250 μs	250 μs	128 Bytes	0-1-3-6	45.12 μs
#2	1	3	250 μs	250 μs	64 Bytes	1-3	21.536 μs
#3	6	1	250 μs	250 μs	32 Bytes	6-3-1	31.024 μs
#4	6	5	250 μs	250 μs	128 Bytes	6-5	23.072 μs
#5	3	0	500 μs	250 μs	64 Bytes	3-1-0	32.048 μs
#6	1	0	500 μs	750 μs	512 Bytes	1-0	32.288 μs
#7	3	6	500 μs	750 μs	32 Bytes	3-6	20.768 μs
#8	3	1	500 μs	500 μs	512 Bytes	3-6-4-2-1	74.576 μs
#9	3	1	500 μs	500 μs	512 Bytes	3-1	32.288 μs
#10	3	1	500 μs	500 μs	1024 Bytes	3-1	44.576 μs
#11	5	3	1 ms	750 μs	64 Bytes	5-4-3	32.048 μs
#12	5	0	1 ms	1 ms	1024 Bytes	5-2-1-0	80.968 μs
#13	1	5	1 ms	2 ms	256 Bytes	1-3-2-5	50.24 μs
#14	0	1	1 ms	5 ms	32 Bytes	0-1	20.768 μs
#15	1	0	2 ms	250 μs	512 Bytes	1-0	32.388 μs
#16	3	0	2 ms	500 μs	512 Bytes	3-1-0	46.384 μs
#17	1	5	2 ms	1 ms	64 Bytes	1-2-5	32.048 μs
#18	0	5	2 ms	2 ms	256 Bytes	0-1-3-4-5	62.288 μs
#19	1	3	2 ms	5 ms	512 Bytes	1-3	32.288 μs

Tabla 5.3: Características de flujos planificados y retardo conseguido tras fallo en enlace

Ahora, las rutas posibles para ambos flujos quedan por tanto limitadas únicamente al camino 0-1 dado que no existe ya ninguna ruta alternativa con la que se pueda comprobar en su gen correspondiente. Con ella aumenta lógicamente el retardo promedio de los flujos desde los 34.2 μs logrados antes hasta los 39.035 μs y con una utilización de los enlaces que pasa del 99.36 % al 99.34 % debido a este caso más desfavorable. El único hueco que se tiene ahora en dicho enlace es el ocasionado por el retraso del flujo #18 para adecuarse a los tiempos del siguiente enlace entre los nodos 1 y 3 dentro del esquema *zero-wait*, tal y como puede verse con la Figura 5.21. Para ello el algoritmo ha estimado oportuno modificar también la ruta del flujo #8, la cual no se veía afectada en principio por el fallo ya que tan solo iba directamente desde el nodo 3 hasta el nodo 1, ampliándola con nuevos saltos adicionales. Si, en cambio, tan solo se modificaran las ruta de estos tres flujos afectados hacia el único enlace que lo permite y con el menor número de saltos a través del ratio 90/10, se lograría evitar tal retardo promedio y mantenerlo en torno a los 36.32 μs, aunque esto reduciría aún más la utilización de los enlaces hasta el 99.3 %, con lo que habría que decidir con los pesos anteriores si conviene más optimizar un parámetro u otro en función de las necesidades como ya se ha visto en el ajuste de los pesos de a función *fitness*. Por ejemplo, en este caso, al contar con un mayor margen de tiempo hasta el retardo límite, podría resultar interesante optimizar la utilización de los enlaces para permitir una mayor tasa de transferencia para el resto de colas.

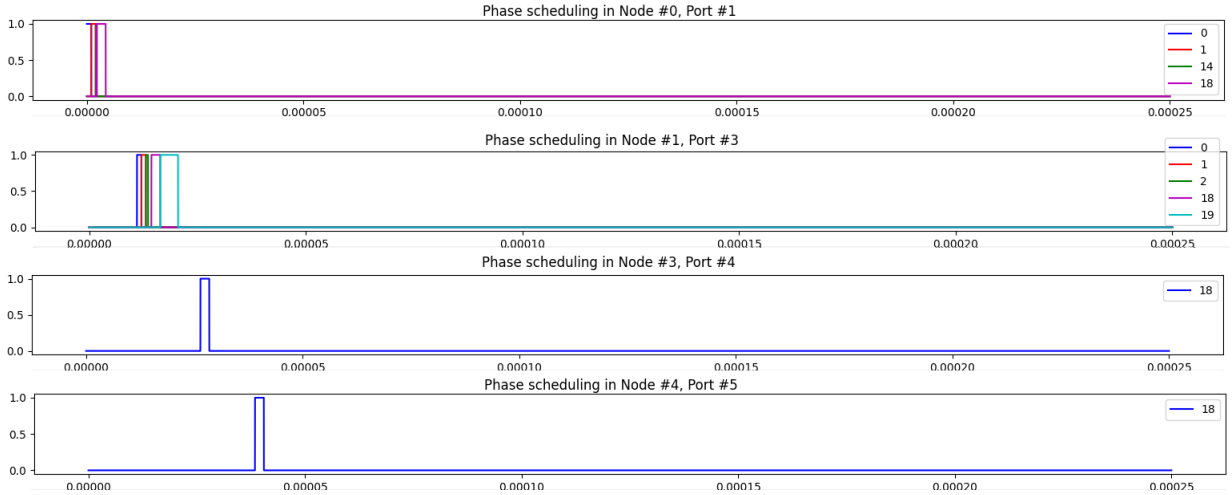


Figura 5.21: Planificación del flujo #18 después del fallo y retraso con *zero-wait*

Finalmente, una vez desarrollado el modelo completo de planificación del TAS y verificado el correcto funcionamiento del mismo así como de la adecuada convergencia del algoritmo genético hacia una solución que optimice parámetros como el retardo extremo a extremo promedio de los flujos y el nivel porcentual de utilización de los enlaces, se puede pasar con la adaptación del mismo con las particularidades de la red inalámbrica 5G anteriormente estudiadas.

5.3. Adaptación del modelo TAS al caso de 5G

Hasta ahora no se ha hablado del *jitter* en el desarrollo del sistema, pues los tiempos se cumplen a lo largo de las rutas formadas por los nodos TSN ya que no existen fluctuaciones en el retardo extremo a extremo debidas al tráfico, pero con 5G y las NFV esto no es

así ya que, al tratarse de servicios virtualizados, el sistema operativo gestiona los recursos hardware en cada tarea de una forma u otra en función de la ejecución del resto de procesos, por lo que se produce el indeterminismo que perjudica al esquema de TSN síncrono. Sin embargo, tecnologías como DPDK pueden llegar a reducirlas considerablemente, aunque permanecen latentes por pequeñas que sean y un sistema síncrono no podría funcionar con ello. En este último apartado de desarrollo se verá la solución propuesta en este proyecto para la absorción de dichas fluctuaciones en el tiempo de procesamiento de las tramas a través de las bandas de guarda con las que cuentan este tipo de redes deterministas.

5.3.1. *Bridge* lógico 5G como caja negra

Si nos adentramos ahora en un caso práctico de la Industria 4.0 con la integración de 5G con redes TSN síncronas nos encontramos con que, tal y como se venía comentando en el Apartado 2.4, existen fluctuaciones producidas por el conjunto de las NFV ubicadas dentro de la red de transporte entre las que se incluyen los RU, DU y CU vistos con anterioridad. Asimismo, como ya se ha visto en la bibliografía con la definición del problema, la mayoría de trabajos consideran la propia red de transporte 5G como una caja negra, es decir, reduciendo la misma a una serie de puertos virtuales que se conectan a otros nodos TSN tal y como se presenta en la Figura 5.22.

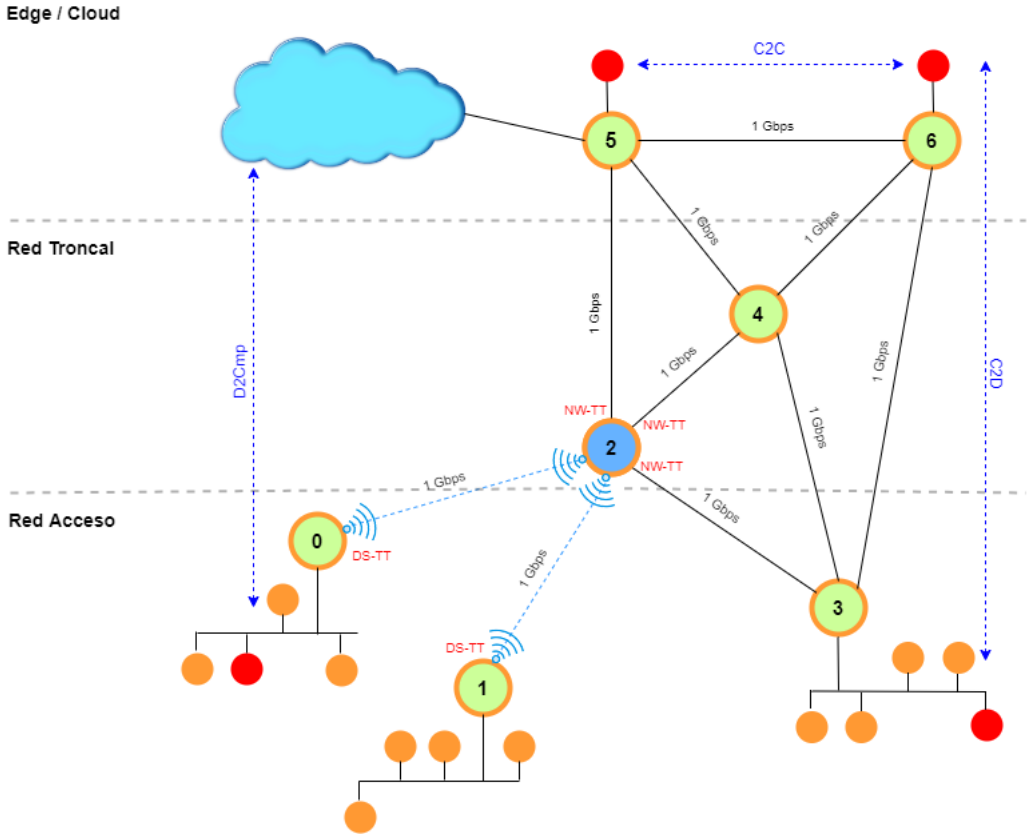


Figura 5.22: Esquema de la nueva topología TSN+5G en la Industria 4.0

Estos puertos virtuales surgen de la interconexión entre ambas tecnologías a través de los traductores DS-TT y NW-TT, aunque también intervienen en ellos las estaciones base gNB y los UPF, respectivamente; por lo que al tratarse de comunicaciones inalámbricas

habrá un retardo que dependerá de las condiciones del canal y con las funciones virtualizadas se producirán ciertas fluctuaciones en el procesamiento de las tramas dentro del mismo plano de datos en la conexión con el resto de la red TSN. Simplificar esto hasta tal punto en el caso de TSN síncrono donde los tiempos de transmisión quedan fijados, bien sea dentro o fuera de la red de transporte de 5G, puede conllevar el riesgo de que el tráfico de los flujos URLLC no se adapte correctamente a los tiempos reservados, incurriendo así en la inestabilidad del sistema ocasionada por el *jitter* al llenarse tales colas prioritarias de tramas que se transmiten fuera de los rangos establecidos e impidiendo el cumplimiento de los requisitos de todos ellos. Por ello se propone el uso de las bandas de guarda que permite emplear TSN a través del GCL de modo que no comience a transmitirse la siguiente trama hasta garantizar que ésta ha completado su transmisión pese a haber podido llegar más tarde del tiempo de inicio que le corresponde en este paradigma *zero-wait*. De este modo se pueden llegar a absorber las fluctuaciones ocasionadas por el *jitter* de las NFV, pero también por el enlace radio dado que los *slices* URLLC se priorizan a nivel de recursos sobre el tráfico como se vio en el Apartado 2.3. De hecho, en algunas publicaciones como la de Bin Hu et al. [48] se llega a hablar de un *jitter* del orden de los 2-5 microsegundos logrados en simulaciones de una red híbrida como en la que se centra este trabajo, aunque para ello es realmente necesaria la sincronización entre las estaciones base gNB y los UE a través de gPTP y con intervalos de sincronización más pequeños para evitar derivas en los relojes. Además, estas bandas de guarda afectan más a la planificación cuanto menor sea el periodo mínimo de fase, pues mayor será el tamaño relativo de éstas y por tanto menor la cantidad de flujos que se podrán planificar. Sin embargo, en este trabajo nos centraremos únicamente en valores estáticos a través del peor caso medido, ya que fijando éste es posible absorber cualquiera de las fluctuaciones a través de dichas bandas de guarda. De este modo, y dado que puede depender tanto de la ruta seguida dentro de la red de transporte de 5G como del tamaño de la trama, la ubicación del UE respecto al gNB, las condiciones del canal, etc; se ha optado por definirse tan solo por un valor porcentual respecto al tamaño de la trama ya que su medición escapa del objeto de este estudio, aunque se ha desarrollado de forma que pueda ser adaptable a cada caso con el simple hecho de modificar los parámetros según convenga. De este modo, dado que los tiempos en la red de transporte 5G quedarían también fijados a lo largo de sus nodos TSN, se podría ahora considerar dicho modelo de caja negra en la que el *bridge* lógico 5G se trataría en su conjunto como una interconexión entre estos puertos virtuales en la que se produce un mayor retardo de procesamiento de las tramas en función del número de saltos y los tiempos planificados en ellos entre gNBs y UPFs; además de la variabilidad que introducen éstos sobre los tiempos de transmisión del último nodo TSN hacia el exterior y que han de mitigarse a través de éste con las bandas de guarda.

5.3.2. Bandas de guarda contra posibles colisiones por fluctuaciones

Estas bandas de guarda deben por tanto aplicarse a los tiempos planificados de aquellos flujos afectados por las fluctuaciones en los nodos que reciban las tramas procesadas por una NFV dentro del *bridge* 5G y deban de transmitirlas a un siguiente nodo. Esto podría ser extrapolable al modelo de caja negra siempre y cuando se ejecuten estas bandas de guarda del mismo modo en su interior, es decir, dentro de la propia red de transporte aunque nos abstraemos de ello; con lo que éste será el encargado de aplicar tal banda de guarda sobre las fluctuaciones con las que se encuentra a la salida de los traductores ya que hasta ese punto los nodos TSN dentro del *bridge* lógico 5G ya habrán absorbido por su parte éstas de la misma manera con el fin de acotar estos tiempos. Por ello, en

este proyecto se ha optado por contemplar una topología más sencilla como la vista en la Figura 5.22, tratando así el modelo de caja negra visto en otras publicaciones pero con la consideración de las bandas de guarda vistas tanto fuera como dentro de este nodo 5G. Una representación de esto puede verse con la Figura 5.23, donde se presenta el mejor de los casos (azul), donde la NFV tarda el menor tiempo posible en procesar la trama; y el peor (rojo), cuando le lleva el mayor tiempo para procesarla. Esto resulta en la banda de guarda (rayado) como diferencia entre el fin de la transmisión de ambos casos. De este modo, y conocido el valor máximo de estas fluctuaciones que transcurre entre t_{11} y t'_{11}, el propio *bridge* lógico 5G ha de establecer un margen de tiempo del mismo tamaño entre t_{12} y t'_{12} como banda de guarda con el fin de evitar colisiones con los siguientes flujos en el caso en que se produjera el peor escenario. Con ello, el nodo TSN al que enlaza el *bridge* 5G comenzaría su transmisión tan pronto como el peor de los casos permitiera, así pudiendo llegar a producirse un hueco hasta el inicio de ésta en t_{21} siguiendo el modelo de planificación llevado a cabo hasta ahora. En cambio, de producirse el mejor caso, el inicio de la transmisión en el siguiente salto de TSN ha de ser considerando siempre el fin de la banda de guarda en el anterior nodo, por lo que de llegar antes tendría inevitablemente que esperar en la cola y acabaría introduciendo un hueco hasta ésta para terminar de acotar el *jitter*, pues de lo contrario se tendrían que estar insertando bandas de guarda a lo largo de la ruta completa del flujo y que que derivaría en un hueco por cada nodo, aunque por ello se podría minimizar el retardo a cambio de introducir *jitter* por toda la red TSN. Otra vía a explorar en este modelo es el uso de *hold-forward* en el traductor para establecer tiempos de espera en los puertos virtuales como se verá más adelante, pues el resultado es el mismo al introducir retardos sobre una cola adicional a las que controla el GCL.

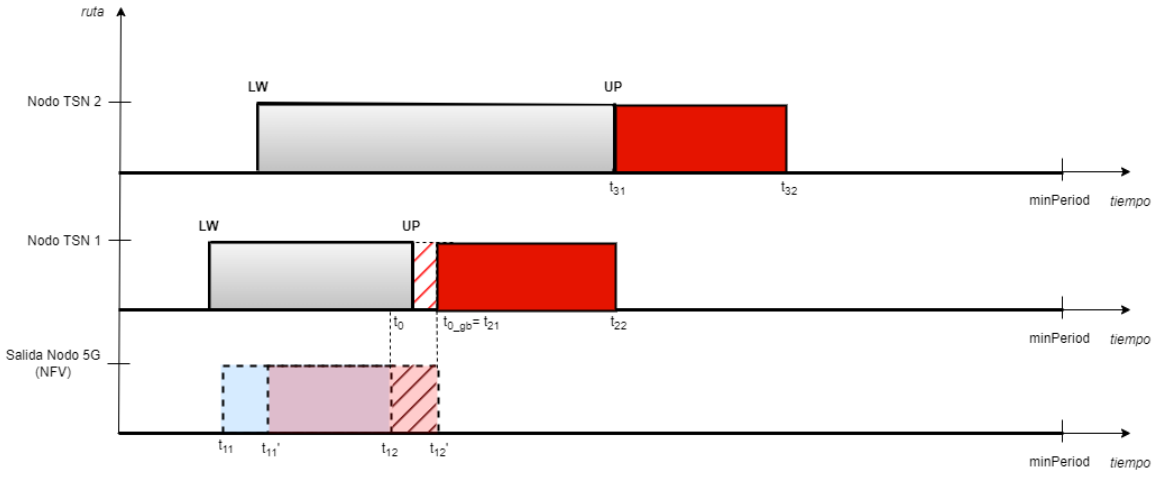


Figura 5.23: Bandas de guarda entre *bridge* 5G y planificación en nodo TSN en única fase

Además, dado que que ahora con el peor caso de estas fluctuaciones podrían producirse incoherencias entre los tiempos de llegada de los diferentes flujos al mismo nodo TSN con el sistema de colas FIFO, en el caso de tener que retrasar este tiempo t_0 como se ha visto con el Algoritmo 3, se podría llegar a provocar la situación que se ve en el primer nodo TSN, donde ahora el tiempo de inicio de la transmisión se ha visto desplazado a un tiempo posterior a UP, con lo que, pese a no ser recomendable el uso de la sentencia *goto*, se ha hecho uso de ella para volver al Caso 1 del Algoritmo 2. Toda la implementación del modelo de integración de 5G y TSN en Python junto con los ficheros de la topología y los flujos de ejemplo puede verse en el repositorio GitHub [49] del proyecto.

Debido al escaso tamaño de estos huecos generados por las bandas de guarda, no se podrá introducir ninguna trama correspondiente a otra cola de prioridad o *best-effort* sobre el mismo intervalo y automáticamente se considerará como hueco a no ser que en éste, después de haber sido el flujo retrasado y aplicando el mismo criterio que con los huecos vistos hasta ahora, se pueda introducir al menos una trama de datos y una banda de guarda del mismo tamaño que evite la colisión con el tráfico de mayor criticidad y se evite en medida de lo posible la continua transición entre colas del GCL según el criterio elegido. Sin embargo, pese a introducir huecos más pequeños y con ello reducir ligeramente la utilización de los enlaces, este sistema podría permitirnos evitar el indeterminismo de los tiempos de procesamiento de las tramas Ethernet de las NFV, lo cual es verdaderamente importante para que TSN síncrono funcione con 5G. Asimismo, se ha optado por crear una nueva matriz en la que se contemple a nivel porcentual esta diferencia entre el mejor caso y el peor respecto al tamaño de la trama. Aquí se tienen en cuenta ahora únicamente tales fluctuaciones en los puertos de salida del *bridge* 5G que conectan con otros nodos TSN, pues se ven afectados por las del UPF, tal y como ya se vio en el Apartado 2.4 con la Figura 3.7. Esta matriz ha de ser por tanto fiel a la topología vista en la Figura 5.22. En este caso se tiene un único *bridge* 5G, correspondiéndose con el nodo 2 y el cual comparte enlaces con los nodos 0, 1, 3, 4 y 5, por lo que se toman los valores que partan de éste hacia ellos y no al revés, salvo que se trate de *half-duplex* y se deban tener en cuenta estos tiempos también en el otro sentido. Estos porcentajes han de adaptarse según la situación de los tiempos medidos sobre las fluctuaciones. A continuación se muestra un ejemplo de ella, donde cada posición corresponde con la transmisión desde el nodo i hasta el nodo j.

$$fluctuationsMatrix \rightarrow link_{i,j}(\%) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 15 & 10 & 0 & 13 & 9 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Finalmente, dado que también se debe considerar el tiempo de residencia en el *bridge* 5G como paso por los nodos TSN que conforman la ruta a través de la red de transporte y de radioacceso, principalmente este último, se ha creado una matriz adicional con las mediciones del retardo entre puertos lógicos y según el camino que lleve un flujos. Nótese que ambos traductores pertenecen al propio *bridge*, por lo que se han de considerar las medidas entre ambos puntos. Es por este motivo por el que dicha matriz corresponde unívocamente al *bridge* 5G del nodo 2 y tendrá como dimensión el número de nodos vecinos con los que conecta, siendo en este caso de 5. De existir otros nodos de tales características, deben tener también su propia matriz, pues de lo contrario sus respectivos valores serán nulos ya que no se estaría introduciendo otro retardo mas que el de la Ecuación 5.1. Por tanto, se ha de anular el valor de t_{proc} para este nodo ya que dentro de esta matriz ya se incluye todo el tiempo de retardo que introducen los nodos de la red de transporte y que transcurre desde un puerto hasta otro, tan solo añadiéndose el intervalo de transmisión en el enlace que lo comunica con un nodo TSN después de haber pasado por los traductores del puerto virtual, tal y como se ha visto en la Figura 5.23. Ahora, el uso de *hold-forward* sí que podría resultar interesante de modo que se retenga una trama con el fin de respetar la banda de guarda entre t_{12} y t'_{12} antes de ser transmitida al siguiente salto de llegar antes en el mejor de los casos, actuando así como una cola adicional. Un ejemplo de ello podría ser la matriz que se muestra a continuación, donde aquellos accesos radio con los nodos

0 y 1 poseen un mayor retardo. Nótese que ahora las filas y columnas se corresponden únicamente con los nodos con los que comparte enlace, bien sea inalámbrico o a través de Ethernet, tratándose así de los nodos 0, 1, 3, 4 y 5, por orden.

$$5GbridgeDelayMatrix \rightarrow ports_{i,j}(\mu s) = \begin{pmatrix} 0 & 40 & 25 & 20 & 25 \\ 40 & 0 & 25 & 25 & 25 \\ 25 & 25 & 0 & 15 & 20 \\ 20 & 25 & 15 & 0 & 15 \\ 25 & 25 & 20 & 15 & 0 \end{pmatrix}$$

Estudiadas ya las características de esta adaptación, podemos pasar a analizar el nuevo modelo a través del algoritmo genético que mejores resultados ha ofrecido.

5.3.3. Resultados de la adaptación del planificador a 5G

Si se ejecuta ahora este nuevo sistema adaptado a las características de 5G, podemos apreciar cómo ahora se introduce un mayor retardo en el nodo 2 debido al número de saltos configurados entre los puertos virtuales y cuya medición se introduce con la anterior matriz, sobre todo entre los nodos 0 y 1 dado que comparten el canal de radioacceso para poder establecer una comunicación con éste. La planificación en torno al *bridge* 5G puede verse en la Tabla 5.4, donde tan solo se realiza una planificación para un total de 20 flujos.

ID	nodo *src*	nodo *dst*	T_{tx}	$Delay_{max}$	*tamaño*	Ruta	$Delay_{e2e}$
#0	0	3	250 μs	250 μs	128 Bytes	0-2-3	49.096 μs
#1	0	6	250 μs	250 μs	128 Bytes	0-2-4-6	55.212 μs
#2	1	3	250 μs	250 μs	64 Bytes	1-2-3	47.048 μs
#3	6	1	250 μs	250 μs	32 Bytes	6-4-3-2-1	66.536 μs
#4	6	5	250 μs	250 μs	128 Bytes	6-5	23.072 μs
#5	3	0	500 μs	250 μs	64 Bytes	3-2-0	47.048 μs
#6	1	0	500 μs	750 μs	512 Bytes	1-2-0	76.384 μs
#7	3	6	500 μs	750 μs	32 Bytes	3-6	20.768 μs
#8	3	1	500 μs	500 μs	512 Bytes	3-4-2-1	75.48 μs
#9	3	1	500 μs	500 μs	512 Bytes	3-2-1	61.384 μs
#10	3	1	500 μs	500 μs	1024 Bytes	3-2-1	77.768 μs
#11	5	3	1 ms	750 μs	64 Bytes	5-6-3	32.048 μs
#12	5	0	1 ms	1 ms	1024 Bytes	5-2-0	77.768 μs
#13	1	5	1 ms	2 ms	256 Bytes	1-2-5	53.192 μs
#14	0	1	1 ms	5 ms	32 Bytes	0-2-1	61.024 μs
#15	1	0	2 ms	250 μs	512 Bytes	1-2-0	76.384 μs
#16	3	0	2 ms	500 μs	512 Bytes	3-2-0	61.384 μs
#17	1	5	2 ms	1 ms	64 Bytes	1-2-5	47.048 μs
#18	0	5	2 ms	2 ms	256 Bytes	0-2-4-5	60.424 μs
#19	1	3	2 ms	5 ms	512 Bytes	1-2-3	61.384 μs

Tabla 5.4: Características de flujos planificados y retardo conseguido en TSN+5G

Como puede verse, los flujos más afectados por el retardo en esta nueva topología son aquellos que hacen uso del enlace radio del nodo 2 para conectar con el resto de la red TSN, especialmente aquellos que tratan de establecer una comunicación entre los dos nodos TSN 1 y 2 que quedan más aislados y que por tanto hacen uso de dicho canal tanto en sentido ascendente como descendente en la misma ruta como es el caso de los flujos #6 y #15. Otros flujos como el #10 y el #12 también ven su retardo incrementado debido a que sus tramas poseen un mayor tamaño y además pasan por el nodo 2 para hacer también uso del canal radio hasta los nodos 1 y 0, respectivamente. Por ende, el retardo extremo a extremo promedio de los flujos asciende hasta los 56.523 μs con una utilización de los enlaces de 99.238 %. Debido a este retardo y como se aprecia en la Figura 5.24, en el caso de tener también que retrasar la transmisión de una trama (e.g. flujo #14) dentro del marco de *zero-wait* se producen huecos aún más considerables en los que, pese a que se podrían llegar a utilizar para otras colas de prioridad, limitan el espacio de la fase disponible para insertar nuevos flujos deterministas, por lo que habría que buscar un algoritmo de planificación más complejo que permitiera reutilizar estos huecos para otros flujos.

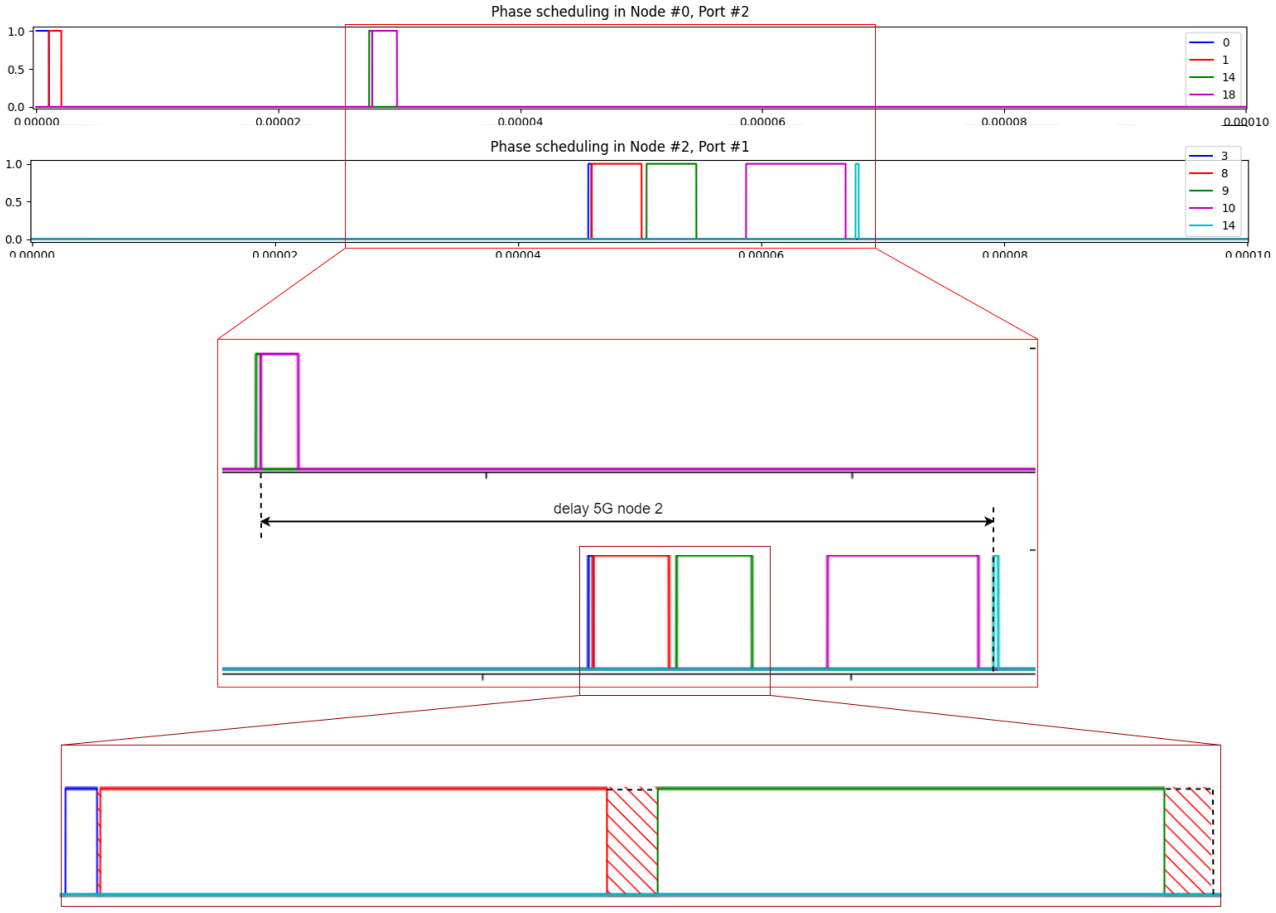


Figura 5.24: Retardo del *bridge* 5G y bandas de guarda en los puertos virtuales de salida

Cabe recordar que a este retardo extremo a extremo de la gráfica hay que añadirle también los tiempos de transmisión entre los dispositivos y los nodos frontera así como el tiempo de procesamiento del último nodo como se vio con la Ecuación 5.2, tratándose así de los tiempos finales que el CUC comunicará a los dispositivos de modo que éstos

configuren sus transmisiones para que cuadren en su llegada con los tiempos establecidos según la topología. Este retardo provocado por el enlace radio se hará aún mayor en la práctica por las condiciones del canal, la distancia hasta la estación base, la capacidad de ésta, etc; pudiendo llegar incluso al milisegundo. Por tanto, el mayor inconveniente de este sistema que modela la planificación de TSN trabajando en conjunto con 5G es que se hace inviable en situaciones en las que aparecen flujos con periodos mínimos tan pequeños como los actuales y con retardos radio de hasta 1 ms, por lo que sería adecuado reajustar el modelo de modo que en cada enlace se tengan en cuenta únicamente los periodos de todos los flujos que hacen uso de él y no como un conjunto completo sobre la red ya que esto permitiría adecuarlo a este caso concreto.

Además, esto también provoca el descarte automático de las soluciones en las que no se pueda cumplir con el tiempo de la fase dado que tan solo con el tiempo entre los puertos lógicos del *bridge* 5G se podría estar ya incumpliendo con esta restricción, por lo que requeriría de un mayor margen de modo que una trama pueda llegar a su destino antes de este límite ya que pasar a la siguiente fase habiéndose planificado ya el primer nodo no respetaría la restricción de las colas FIFO. Esto se produce sobre todo a medida que se aumenta el volumen de flujos a planificar, con lo que este modelo es más adecuado en el caso de un menor número de flujos en el caso de integrarlo con 5G.

Adicionalmente, para evitar la aparición de huecos excesivamente grandes que limitaran la capacidad de planificar nuevos flujos al no poder ser utilizados, se podría también penalizar dicho tamaño pese a que en él pudieran insertarse otras tramas del resto de colas. Futuras versiones de este sistema se centrarán en la mejora de su comportamiento con el objetivo de lograr una mayor optimización.

Capítulo 6

Conclusiones y líneas futuras

Para concluir con este trabajo, se ha estudiado en profundidad el uso de tecnologías como TSN y 5G para comprender mejor sus características así como el estado actual de su integración en un escenario industrial, más concretamente el de la actual Industria 4.0 donde impera la necesidad de automatizar los procesos con el fin de optimizar su eficiencia así como de dotar de flexibilidad inalámbrica a los dispositivos dado el mayor volumen de comunicaciones entre sensores y controladores con un menor coste de despliegue asociado a sus enlaces. Además, existen aplicaciones en las que se ha de velar por la sincronización entre dispositivos con el fin de garantizar un retardo y un *jitter* por debajo de un determinado umbral, como puede ser el caso de *platooning* para mejorar la eficiencia entre el consumo de combustible y la velocidad de un convoy de vehículos autónomos que comparten información periódicamente. Con ello, se ha logrado percibir la complejidad de dicha integración entre estas dos tecnologías debida principalmente al problema del indeterminismo introducido por las variaciones del canal radio y la gestión de los procesos en las NFV de la red de transporte de 5G con el caso de un esquema de TSN síncrono, donde los tiempos se definen de forma fija a través de una planificación de los distintos flujos que hacen uso de la red. Sin embargo, se han visto algunas publicaciones en las que se presentan posibles vías para mitigar estos efectos como puede ser el uso de modelos matemáticos en la asignación de los recursos radio o la gestión del hardware de los servidores a través de DPDK; además de la propuesta propia de este trabajo del uso de bandas de guarda con el fin de absorber estas fluctuaciones en el procesamiento de las tramas. Se han conocido también otras publicaciones en las que se propone una integración entre los diferentes bloques lógicos que controlan y coordinan los recursos entre ambas tecnologías, aunque si bien es cierto que en algunos casos existen ciertas disconformidades entre ellas al tratarse de un trabajo muy actual en progreso en el que no existe una estandarización acerca de la integración. Por tanto, este trabajo se ha centrado principalmente en la investigación sobre el estado del arte de estos aspectos junto con la propuesta de optimización del planificador a través de la metaheurística mediante un algoritmo genético en un escenario real a través de un modelo y teniendo en cuenta las características de este problema, por lo que se trata de un proyecto más orientado a la vanguardia de esta integración.

Para ello, se han analizado los métodos de planificación de TSN síncrono más tradicionales basados en la heurística o en la optimización lineal, como es el caso de MILP; así como aquellos más avanzados que hacen uso de la Inteligencia Artificial para su resolución, más concretamente aquellos centrados en el aprendizaje automático a través de *Reinforcement Learning* y algoritmos genéticos, optando por éstos últimos dada su menor

complejidad para la resolución y que se trata de seleccionar la ruta adecuada para cada flujo en un paradigma de *zero-wait*, con lo que esta vía es idónea. De esta forma, se ha desarrollado un modelo TAS más sencillo en Python basado en las restricciones impuestas por los tiempos de llegada de las tramas de los flujos a cada nodo TSN con el fin de evitar colisiones en un sistema de colas de prioridad FIFO y, a partir de ello, ejecutar un algoritmo genético mediante la librería PyGAD que evalúe tanto el retardo extremo a extremo de los distintos flujos, influido principalmente por el número de saltos; como la utilización de los enlaces en cuanto a la aparición de huecos en los que no se puede transmitir una sola trama de cualquiera de las colas a través de las diferentes combinaciones de las rutas seguidas por éstos dentro de un espacio de búsqueda acotado y codificadas a través de los genes de un individuo de la población.

Una vez desarrollado el código necesario para implementar el modelo, se han comprobado las distintas combinaciones que ofrece la librería para la selección y el cruce entre individuos con el fin de lograr mejores resultados en lo que a valor *fitness* y tiempo de convergencia se refiere, pues se ha visto que algunas de ellas empeoraban a medida que se incrementa la presión selectiva con métodos como el de selección por ruleta o incluso cuando ésta se aleatoriza, sobre todo a medida que se aumentaba el número de flujos a planificar, concluyendo así con que era más adecuado emplear el método de selección basada en ranking con una probabilidad de mutación de los genes relativamente baja, en torno al 5-10 %.

Sin embargo, se ha advertido la limitación de los algoritmos genéticos en la convergencia hacia una solución pues ésta puede variar en distintas ejecuciones dada la aleatoriedad en las distintas etapas del proceso cíclico así como en la inicialización de la población, con lo que se deben llevar a cabo varias ejecuciones para extraer aquella con mayor *fitness*. Otro inconveniente de ello es la compensación entre el espacio de búsqueda, el límite en las generaciones y el tiempo de cómputo para lograr la mejor solución posible. Al poder contar con el supercomputador de la Universidad de Granada, se ha establecido un límite de 15.000 generaciones, acotado a 5.000 si no se producen cambios del valor *fitness* durante éstas; además de una población fijada a 100 individuos un espacio de búsqueda con tan solo las 6 rutas con menor número de saltos para cada flujo.

Adicionalmente, se ha comprobado el rendimiento del sistema a través de otros aspectos como la convergencia hacia una solución u otra en función de los pesos, la respuesta para el retardo y los huecos según el volumen de flujos a planificar o la respuesta ante posibles fallos después de haber seleccionado de forma *offline* la mejor solución para cada situación, demostrando así un buen funcionamiento de éste. No obstante, se ha visto a través de los pesos la complejidad en el control sobre los huecos al producirse tales oscilaciones en los resultados, por lo que ha sido necesario ejecutar múltiples veces el sistema para conseguir el caso más favorable respecto al *fitness*; aunque la posibilidad de controlar la optimización de un parámetro u otro a través de estos pesos es clara. También se ha visto una tendencia al alza del retardo y a los huecos a medida que se incrementa el número de flujos cuando ambos pesos son similares, pues el modelo trata en un compromiso entre ambos parámetros de equilibrar el mayor número de huecos que aparece a costa de sacrificar el retardo conseguido. Sin embargo, el coste computacional del algoritmo genético se prevé que sea muy inferior al que tendría *Reinforcement Learning* a través de redes neuronales y pequeños cambios en cada iteración, mientras que aquí se logra mejorar el *fitness* considerablemente en escasas generaciones, sobre todo en el caso de elevar el número de flujos planificados y contar con un mayor espacio de búsqueda. Además, el sistema habría sido capaz de recuperarse después de haberse detectado un fallo en un enlace de modo que, en función

de la nueva topología resultante, se establecerían unos nuevos tiempos para cada uno de los flujos de modo que no solo cumplan sus restricciones sino que además lo hagan de una forma más óptima, aunque se favorecerá más el retardo extremo a extremo dado uso de *zero-wait*.

Por último, se ha probado a implementar la solución propuesta para paliar las fluctuaciones ocasionadas por las NFV sobre la red TSN síncrona tanto dentro de la red de transporte como fuera a través de las bandas de guarda en un modelo de *bridge* 5G como caja negra, en la que se produce un retardo medido entre sus puertos virtuales al que hay que añadirle un intervalo de espera de modo que absorba el *jitter* producido por estas funciones virtualizadas. Sin embargo, debido al mayor retardo que introduce el enlace radio, el cual puede llegar hasta 1 ms, dificulta la utilidad de este modelo cuando el periodo mínimo de todos los flujos es inferior a este tiempo.

Finalmente, como líneas futuras de trabajo se propone la continuación en el desarrollo de este modelo dentro del proyecto 6G-CHRONOS a través de un sistema de planificación más inteligente, en el que los márgenes de fase puedan intervenir en cada una de ellas sobre la población del algoritmo genético, evitando así en todo momento el desbordamiento para favorecer la planificación de un mayor volumen de flujos y dotar así de mayor inteligencia en la selección de las fases, pues a priori no se puede conocer a ciencia cierta un valor del todo adecuado que permita salvar las rutas con mayor número de saltos. También se pretende investigar sobre vías alternativas a *zero-wait* para la minimización conjunta del retardo y de los huecos así como de otros métodos más complejos para el algoritmo genético. Además, dado que en algunos casos se tratan de comunicaciones bidireccionales, se pretende implementar para cada flujo un camino de retorno transcurrido un tiempo de cómputo en el PLC previamente medido. Asimismo, se pretende probar este modelo sobre un escenario real formado por equipos reales de conmutación compatibles con TSN como los *White Rabbit Z16*, por lo que para ello habrá que ajustar los parámetros como el tiempo de procesamiento de cada nodo, medir el retardo de propagación, etc. Por último, se tratará de paralelizar la ejecución del código que implementa el modelo completo para minimizar en lo posible el tiempo de cómputo, pues la vía que ofrece PyGAD de *multithreading* acarrea problemas con la interdependencia entre las variables de cada solución. Con todo ello, se tratará de profundizar más sobre las fluctuaciones introducidas por las NFV y estudiar su viabilidad de forma experimental, con un sistema de planificación a través de una división de fases independiente para cada enlace.

No obstante, con este trabajo se han logrado comprender las nociones básicas acerca de la integración entre ambas tecnologías orientada a redes deterministas en el ámbito de la Industria 4.0, llegando así a reconocer las grandes diferencias que poseen respecto a las redes de tráfico de datos más tradicionales vistas a lo largo de la carrera de Ingeniería de Telecomunicación; así como a poner los conocimientos adquiridos en práctica para resolver las problemáticas que suponen con un modelo de planificación funcional como el desarrollado. Asimismo, se considera un gran avance en el estudio de la Inteligencia Artificial aplicada a las telecomunicaciones fuera de la tendencia actual de usar únicamente técnicas de *Deep Learning* en cualquier campo, pues se deben barajar también técnicas alternativas como puede ser el caso de los algoritmos genéticos para la resolución de problemas de este tipo.

Glosario

3GPP	3rd Generation Partnership Project.
5G-ACIA	5G Alliance for Connected Industries and Automation.
5G-PPP	5G Infrastructure Public-Private Partnership.
5QI	5G QoS Indicator.
AF	Application Function.
AMF	Access and Mobility-management Function.
API	Application Programming Interface.
ATS	Asyncrhonous Traffic Shaper.
BMCA	Best Master Clock Algorithm.
C2C	Controller-to-Controller Communication.
C2D	Controller-to-Device Communication.
CM	Clock Master.
CNC	Centralized Network Controller.
CPS	Cyber-Physical System.
CRS	Cell-specific Reference Signal.
CS	Clock Slave.
CSMF	Communication Service Management Function.
CU	Centralized Unit.
CUC	Centralized User Configuration.
D2Cmp	Device-to-Compute Communication.
DFS	Depth-First Search.
DRL	Deep Reinforcement Learning.
DRLS	Deep Reinforcement Learning-based Scheduler.
DS-TT	Device-side TSN Translator.
DU	Distributed Unit.
eCPRI	evolved Common Public Radio Interface.
EDF	Earliest Deadline First.
eMBB	enhanced Mobile BroadBand.
FDMA	Frequency-Division Multiple Access.
FIFO	First Input First Output.
FRER	Frame Replication & Elimination Reliability.

GCL	Gate Control List.
GM	Grand-Master.
gPTP	generic Precision Time Protocol.
GTP-U	GPRS Tunneling Protocol - User Plane.
HARQ	Hybrid Automatic Repeat reQuest.
HLS	Heuristic List Scheduler.
HMI	Human-Machine Interface.
IEEE	Institute of Electrical and Electronic Engineers.
IIoT	Industrial Internet of Things.
IMT	International Mobile Telecommunications.
IoT	Internet of Things.
IS-IS	Intermediate System to Intermediate System.
ITU	International Telecommunication Union.
LAG	Link Aggregation Group.
LD	Low Degree.
LDPC	Low Density Parity Check.
LLC	Link Layer Control.
LLDP	Link Layer Discovery Protocol.
LRP	Link-local Registration Protocol.
LS	List Scheduler.
LW	Lower Bound.
M2M	Machine-to-Machine.
MAC	Medium Access Control.
MCD	Máximo Común Divisor.
MCM	Mínimo Común Múltiplo.
MILP	Mixed-Integer Linear Programming.
mMIMO	Massive Multiple Input Multiple Output.
mMTC	Massive Machine-Type Communications.
MNO	Mobile Network Operator.
MTNSC	Multilayer Transport Network Slice Controller.
MTU	Maximum Transmission Unit.
NAS	Non-Access Stratum.
NEF	Network Exposure Function.
NFV	Network Function Virtualization.
NR	New Radio.
NSI	Network Slice Instance.
NSMF	Network Slice Management Function.
NSSF	Network Slice Selection Function.
NSSI	Network Slice Subnet Instance.
NSSMF	Network Slice Subnet Management Function.
NW-TT	Network-side TSN Translator.
O-RAN	Open Radio Access Network.
OFDM	Orthogonal Frequency Division Multiplexing.

OMT	Optimization Modulo Theories.
PCE	Path Computation Element.
PCF	Policy Control Function.
PCP	Priority Code Point.
PCR	Path Control and Reservation.
PDCCH	Physical Downlink Control Channel.
PDCP	Packet Data Convergence Protocol.
PDSCH	Physical Downlink Shared Channel.
PDU	Protocol Data Unit.
PLC	Programmable Logic Controller.
PRACH	Physical Random Access Channel.
PSFP	Per-Stream Filtering and Policing.
PTSP	Pseudo Time Synchronization Packet.
PUCCH	Physical Uplink Control Channel.
PUSCH	Physical Uplink Shared Channel.
QoS	Quality of Service.
RB	Resource Block.
RE	Resource Element.
RLC	Radio Link Control.
RRC	Radio Resource Control.
RSTP	Rapid Spanning Tree Protocol.
RU	Radio Unit.
SBA	Service-Based Architecture.
SCTP	Stream Control Transmission Protocol.
SDAP	Service Data Adaptation Protocol.
SDN	Software-Defined Network.
SFTP	SSH File Transfer Protocol.
SINR	Signal-to-Interference-plus-Noise Ratio.
SLURM	Simple Linux Utility for Resources Management.
SMF	Session Management Function.
SMSF	Short Message Service Function.
SMT	Satisfiability Modulo Theories.
SNMP	Simple Network Management Protocol.
SPB	Shortest Path Bridging.
SRP	Stream Reservation Protocol.
TAS	Time-Aware Shaper.
TDMA	Time-Division Multiple Access.
TSN	Time-Sensitive Networking.
UBS	Urgency-Based Scheduler.
UDM	Unified Data Management.
UDR	Unified Data Repository.
UDSF	Unstructured Data Storage Function.
UE	User Equipment.

UNI	User-Network Interface.
UP	Upper Bound.
UPF	User Plane Function.
URLLC	Ultra-Reliable Low-Latency Communications.
VLAN	Virtual Local Area Network.
WSN	Wireless Sensor Network.

Bibliografía

[1] Sriganesh K. Rao y Ramjee Prasad. Impact of 5G Technologies on Industry 4.0. *Wireless Pers Commun*, pages 145–159, marzo 2018.

[2] Zexian Li, Mikko A. Uusitalo, Hamidreza Shariatmadari, and Bikramjit Singh. 5G URLLC: Design Challenges and System Concepts. In *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6. 2018.

[3] Sangkyu Baek, Donggun Kim, Milos Tesanovic, and Anil Agiwal. 3GPP New Radio Release 16: Evolution of 5G for Industrial Internet of Things. *IEEE Communications Magazine*, 59(1):41–47, 2021.

[4] Sushmit Bhattacharjee, Kostas Katsalis, Osama Arouk, Robert Schmidt, Tongtong Wang, Xueli An, Thomas Bauschert, and Navid Nikaein. Network Slicing for TSN-Based Transport Networks. *IEEE Access*, 9:62788–62809, 2021.

[5] 5G Alliance for Connected Industries and Automation. *5G-ACIA*. `https://5g-acia.org/`, último acceso: 09/03/22.

[6] Francisco Manuel Pozo Pérez. *Methods for Efficient and Adaptive Scheduling of Next-Generation Time-Triggered Networks*. PhD thesis, Mälardalen University, 2019.

[7] Yang Wang, Jidong Chen, Wei Ning, Hao Yu, Shimei Lin, Zhidong Wang, Guanshi Pang, and Chao Chen. A Time-Sensitive Network scheduling algorithm based on Improved Ant Colony Optimization. *Alexandria Engineering Journal*, 60(1):107–114, 2021.

[8] Borja Bordel Sánchez, Ramón Pablo Alcarria Garrido, Diego Sánchez de Rivera, and Álvaro Sánchez Picot. Enhancing process control in Industry 4.0 scenarios using Cyber-Physical Systems. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 7:41–64, 2016.

[9] i-SCOOP. Industry 4.0 and the fourth industrial revolution explained. `https://www.i-scoop.eu/industry-4-0/`, último acceso: 16/03/22.

[10] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. The Industrial Internet of Things (IIoT): An analysis framework. *Computers in industry*, 101:1–12, 2018.

[11] Real Time Innovations Inc, Industrial Internet of Things. *RTI FAQ*, diciembre 2015. `https://info.rti.com/hubfs/docs/Industrial_IoT_FAQ.pdf`, último acceso: 10/03/22.

[12] Sita Rani, Aman Kataria, and Meetali Chauhan. Fog computing in industry 4.0: Applications and challenges—a research roadmap. In *Energy Conservation Solutions for Fog-Edge Computing Paradigms*, pages 173–190. Springer, 2022.

[13] TS 22.104 Service requirements for cyber-physical control applications in vertical domains (v18.3.0). *3GPP Technical specification*, diciembre 2021. `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3528`, último acceso: 10/03/22.

[14] IEEE 1588-2019 Evolves to Better Serve its Wide Variety of Applications. *IEEE SA*. `https://sagroups.ieee.org/1588/news/ieee-1588-2019-evolves-to-better-serve-its-wide-variety-of-applications/`, último acceso: 21/03/22.

[15] Time-Sensitive Networking (TSN) Task Group. *IEEE 802.1 Task Group*. `https://1.ieee802.org/tsn/`, último acceso: 21/03/22.

[16] 802.1AB-2016 - IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery. *IEEE SA*. `https://ieeexplore.ieee.org/document/7433915`, último acceso: 21/03/22.

[17] Ahmed Nasrallah, Venkatraman Balasubramanian, Akhilesh Thyagaturu, Martin Reisslein, and Hesham ElBakoury. Reconfiguration algorithms for high precision communications in Time Sensitive Networks: Time-Aware Shaper configuration with IEEE 802.1Qcc. *ITU*, 2019. `https://www.itu.int/pub/S-JNL-VOL2.ISSUE1-2021-A02`, último acceso: 21/03/22.

[18] Harri Holma, Antti Toskala, and Takehiro Nakamura. *5G technology: 3GPP New Radio*. John Wiley & Sons, 2020.

[19] M. Series. IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU*, 2083:21, 2015.

[20] George Yammine, Mohammad Alawieh, Gregor Ilin, Mohammad Momani, Mostafa Elkhouly, Piotr Karbownik, Norbert Franke, and Ernst Eberlein. Experimental Investigation of 5G Positioning Performance Using a mmWave Measurement Setup. In *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2021.

[21] Jose Luis Carcel, Belkacem Mouhouche, Manuel Fuentes, Eduardo Garro, and David Gomez-Barquero. IMT-2020 key performance indicators: Evaluation and extension towards 5G new radio point-to-multipoint. In *2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5. IEEE, 2019.

[22] TS 23.501 System architecture for the 5G system (v17.4.0). *3GPP Technical specification*, marzo 2022. `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144`, último acceso: 05/04/22.

[23] 5G End-to-end Network Slice Mapping from the view of Transport Network. *IETF*. `https://tools.ietf.org/id/draft-geng-teas-network-slice-mapping-02.html`, último acceso: 05/04/22.

[24] TS 24.535 Device-Side Time Sensitive Networking (TSN) Translator (DS-TT) to Network-Side TSN Translator (NW-TT) protocol aspects (v17.2.0). *3GPP Technical specification*, enero 2022. `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3680`, último acceso: 10/04/22.

[25] Janos Farkas, B Varga, G Miklós, and J Sachs. 5G-TSN integration meets networking requirements for industrial automation. *Ericsson: Stockholm, Sweden*, pages 0014–0171, 2019.

[26] Michail-Alexandros Kourtis, Andreas Oikonomakis, Dimitris Santorinaios, Themis Anagnostopoulos, Giorgios Xilouris, Anastasios Kourtis, Ioannis Chochliouros, and Charilaos Zarakovitis. 5G NPN Performance Evaluation for I4. 0 Environments. *Applied Sciences*, 12(15):7891, 2022.

[27] Report describing the framework for 5G systems and Network Management Functions. *5G-SMART European Project*. `https://5gsmart.eu/wp-content/uploads/5G-SMART-D5.5-v1.0.pdf`, último acceso: 05/04/22.

[28] David Ginthör, René Guillaume, Maximilian Schüngel, and Hans D Schotten. Robust End-to-End Schedules for Wireless Time-Sensitive Networks under Correlated Large-scale Fading. In *2021 17th IEEE International Conference on Factory Communication Systems (WFCS)*, pages 115–122. IEEE, 2021.

[29] David Ginthör, René Guillaume, Maximilian Schüngel, and Hans D Schotten. 5G RAN Slicing for Deterministic Traffic. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2021.

[30] Yajing Zhang, Qimin Xu, Mingyan Li, Cailian Chen, and Xinping Guan. QoS-Aware Mapping and Scheduling for Virtual Network Functions in Industrial 5G-TSN Network. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[31] Hao Zhou, Medhat Elsayed, and Melike Erol-Kantarci. RAN Resource Slicing in 5G Using Multi-Agent Correlated Q-Learning. In *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1179–1184. IEEE, 2021.

[32] Jonathan Falk, Frank Dürr, and Kurt Rothermel. Exploring Practical Limitations of Joint Routing and Scheduling for TSN with ILP. In *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 136–146, 2018.

[33] David Hellmanns, Lucas Haug, Moritz Hildebrand, Frank Dürr, Stephan Kehrer, and René Hummen. How to optimize joint routing and scheduling models for TSN using Integer Linear Programming. In *29th International Conference on Real-Time Networks and Systems*, pages 100–111, 2021.

[34] Silviu S Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 183–192, 2016.

[35] Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. Heuristic list scheduler for time triggered traffic in Time Sensitive Networks. *ACM Sigbed Review*, 16(1):15–20, 2019.

[36] Frank Dürr and Naresh Ganesh Nayak. No-wait packet scheduling for IEEE Time-Sensitive Networks (TSN). In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 203–212, 2016.

[37] Florian Heilmann and Gerhard Fohler. Size-based queuing: An approach to improve bandwidth utilization in TSN networks. *ACM SIGBED Review*, 16(1):9–14, 2019.

[38] Francisco Pozo, Guillermo Rodriguez-Navas, and Hans Hansson. Schedule Reparability: Enhancing Time-Triggered Network Recovery Upon Link Failures. In *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 147–156, 2018.

[39] Chunmeng Zhong, Hongyu Jia, Hai Wan, and Xibin Zhao. DRLS: A Deep Reinforcement Learning Based Scheduler for Time-Triggered Ethernet. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–11. IEEE, 2021.

[40] Hongyu Jia, Yu Jiang, Chunmeng Zhong, Hai Wan, and Xibin Zhao. TTDeep: Time-Triggered Scheduling for Real-Time Ethernet via Deep Reinforcement Learning. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[41] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[42] Maryam Pahlevan and Roman Obermaisser. Genetic algorithm for scheduling time-triggered traffic in Time-Sensitive Networks. 1:337–344, 2018.

[43] Ou Xinjian, Liao Jingjing, Chen Chaofeng, You Zilin, Li Xiang, and Hu Shukai. Research on 5G Microservices Capability Open Architecture and Deterministic Bearing Technology. In *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pages 492–496, 2021.

[44] Charalampos Stylianopoulos, Magnus Almgren, Olaf Landsiedel, Marina Papatriantafilou, Trevor Neish, Linus Gillander, Bengt Johansson, and Staffan Bonnier. On the performance of commodity hardware for low latency and low jitter packet processing. In *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems*, pages 177–182, 2020.

[45] Ahmed Lahjouji El Idrissi, Chakir Tajani, and Mohammed Sabbane. New crossover operator for genetic algorithm to resolve the fixed charge transportation problem. *Journal of Theoretical & Applied Information Technology*, 95(8), 2017.

[46] Ahmed Fawzy Gad. PyGAD: An Intuitive Genetic Algorithm Python Library, 2021.

[47] OMNeT++. `https://omnetpp.org/`, último acceso: 09/03/22.

[48] Bin Hu and Hamid Gharavi. A hybrid wired/wireless deterministic network for smart grid. *IEEE Wireless Communications*, 28(3):138–143, 2021.

[49] Pablo Rodríguez Martín. Model for 5G network slicing scheduling with TSN. `https://github.com/paroma96/TSN_5G_integration`, septiembre 2022.