# Proyecto

*DE*

# *DISEÑO Y DESARROLLO DE HERRAMIENTAS DIDÁCTICAS PARA EL ANÁLISIS DEL RENDIMIENTO DE REDES*

Documento **Memoria**

**Autor** Sayans Alberdi, Ainhoa

**Director** Ferro Vázquez, Armando

**Fecha** Septiembre 2016

# 1 ÍNDICE

# 2 LISTA DE TABLAS/ILLUSTRACIONES/ACRONIMOS

## 2.1 Índice de ilustraciones



## 2.2 Índice de tablas

# 3 RESUMEN TRILINGÜE

## 3.1 Castellano

En la asignatura Rendimiento en Redes de Telecomunicación se ha detectado la necesidad de simplificar el aprendizaje de las herramientas utilizadas. Ante la situación se procede a la realización de diversas librerías que permitan acelerar la familiarización del alumnado con los diferentes entornos de desarrollo. Junto con las bibliotecas generadas se deberá entregar el manual de las mismas. En el presente documento se analizarán los pasos llevados a cabo para la realización del proyecto de simplificación del aprendizaje.

## 3.2 Inglés

On the subject Performance in Telecommunication Networks has been identified the need to simplify the learning of the tools used. Taken the situation in to account we proceed to perform various libraries to accelerate the familiarization of students with the different development environments. Along with the generated libraries is to be delivered the correspondent manual. In this document will be discussed the steps taken for the project to simplify the learning.

## 3.3 Euskera

Telekomunikazio sareen errendimendua irakasgaian erabilitako erraminten ikasketa sinplifikatzeko beharra antzeman da. Egoeraren aurrean garapen-inguruen desberdinekin familiarizazioa bizkortze baimentzen duten liburutegia batzuen garapenari ekiten zaio. Sortutako liburutegiekin batera berei buruzko eskuliburua ema behar izango da. Dokumentu honetan ikasketaren sinplifikazio proiektuaren errealizaziorako burututako pausako analizatuko dira.

Keyword: librerías, aprendizaje, herramientas

# 4 INTRODUCCIÓN

Desde hace años los avances en la comunicación informática están revolucionando la sociedad. La capacidad de todo tipo de dispositivos y plataformas para conectarse entre sí mediante la red global ha generado que las personas estén en todo momento en contacto con otros usuarios y con los diferentes equipos. Esta capacidad ha llevado a la sociedad a informatizar todo tipo de elementos y tareas, este hecho es la razón de la denominación era digital de este periodo.

La era digital ha generado un mayor impacto en las empresas que en las personas individuales. Los distintos negocios almacenan información y recursos vitales de los mismos en dispositivos informáticos o en la nube. El hecho de almacenar datos y herramientas de manera digital genera la necesidad de tener acceso a las mismas en todo momento. Ante esta situación el funcionamiento de las distintas organizaciones está ligado al rendimiento de sus redes y equipos. Esta relación implica que tanto el equipamiento digital como la comunicación entre los mismo debe estar disponible en todo momento.

La necesidad de interconexión continua entre los distintos elementos de una organización implica que la red tiene que estar debidamente establecida y monitorizada. El control de la red debe ser online e interferir en la menor medida en el funcionamiento de la misma. En caso de ser necesaria una modificación o ampliación de los servicios prestados por la red de la organización se deberá realizar el pertinente estudio y planificación para su correcto funcionamiento.

La planificación debe comenzar por la especificación del tipo de tráfico que cursará los equipos de red y el tipo de trabajo que realizarán los servidores. Con este conocimiento se podrá modelar un sistema matemático que emule el funcionamiento permitiendo detectar posible cuellos de botella o limitaciones que presente. Tras esto se deberá realizar una simulación más completa de la estructura diseñada para validar el funcionamiento.

Actualmente existen diversos simuladores de red que permiten comprobar que los diseños generados son adecuados según la inserción de tráfico establecida. En el transcurso del Master en Ingeniería de Telecomunicación se hace uso de varios de estos simuladores con el objetivo evaluar diversos protocolos y estándares. Por el contrario, la asignatura Rendimiento en Redes de Telecomunicación (RRT)[1] se orienta en las primeras fases de la planificación con el modelado y las simulaciones básicas. El objetivo de la asignatura es que el alumno adquiera el conocimiento básico del modelado matemático de redes y servicios para el estudio del rendimiento de protocolos y sistemas de telecomunicaciones.

Las herramientas utilizadas en la asignatura, Mathematica[3] y Omnet++[2], requieren de cierto conocimiento básico para el desarrollo de experimentos que permitan evaluar la teoría de colas o el rendimiento de trasmisiones ARQ, temas tratados en la asignatura. El tiempo necesario para hacer uso de las herramientas correctamente ralentiza el avance de las prácticas disminuyendo la cantidad de pruebas que se pueden realizar. Ante la problemática de la curva de aprendizaje de estos entornos de desarrollo, es necesario la realización de librerías de apoyo que permitan facilitar el proceso de aprendizaje en los entornos docentes.

En el presente documento se razonarán y analizarán las librerías desarrolladas dentro del proyecto "Diseño y desarrollo de herramientas didácticas para el análisis del rendimiento de redes". Se expondrán los objetivos del proyecto, la metodología utilizada para llevarlos a cabo, las distintas alternativas de diseño valoradas así como la seleccionada y la solución final y considerada más óptima para cumplir las especificaciones.

# 5 ESTADO DEL ARTE

En la presente sección se analizará la línea histórica y la situación actual de las diversas temáticas que se abordan en el proyecto. De manera independiente se quiere identificar la capacidad de los distintos entornos de desarrollo para dar solución a dichas problemáticas.

## 5.1 Situación de las temáticas

Los primeros pasos de la comunicación entre dos dispositivos fue la red telefónica. Esta red estaba basada en la conmutación de circuitos. Con el objetivo de dimensionar correctamente las líneas y centrales de conmutación telefónicas el ingeniero Agner Kraup Erlang[4] en 1909 publicó el primer artículo sobre teoría de colas. Este artículo se centraba en un estudio matemático de sistemas de simulación para el análisis del comportamiento de las líneas telefónicas de espera. Esta teoría se ha utilizado en diversos ámbitos de la sociedad. En 1953 David G. Kendall[5] introdujo la notación A/B/C que más adelante fue extendida a A/B/c/K/m/Z

1. Código que describe el proceso de llegada

   M – Markoviana, distribución exponencial de los tiempos entre llegadas

   $E_K$ – Erlang de k etapas

   $H_K$ – Hiperexponencial de k etapas

   D – Deterministas

   G – General

2. Código que describe el proceso de servicio. Mismo símbolos que para el proceso de llegadas
3. Número de servidores que atienden las peticiones recibidas
4. Número máximo de clientes permitidos en el sistema incluyendo aquellos a los que se está prestando servicio
5. Tamaño de la fuente
6. Disciplina de la cola

   FIFO – First In First Out

   LIFO – Last In First Out

   SIRO – Service in Random Order

En la década de los 60 se comenzó a trabajar en la conexión de diversos equipos de tal manera que se pudiera compartir información y computación en sistemas distribuidos. El tráfico creado por este tipo de uso se desarrolla en ráfagas, intervalos de actividad seguidos de periodos de inactividad. Ante esta situación diversos grupos de investigación

de manera independiente comenzaron el desarrollo del concepto de conmutación de paquetes como alternativa a la ya utilizada y generalizada conmutación de circuitos. Durante este tiempo se hizo uso de la teoría de colas para demostrar que la aproximación de conmutación de paquetes era más adecuada para el tráfico a ráfagas.

Durante la década de los 70 y con la irrupción de Ethernet se extendieron las redes en LAN. La extensión de este tipo de redes fomentó el desarrollo de proyectos para la interconexión de redes. En estos proyectos derivaron en la generación en las primeras versiones del protocolo TCP que incluían muchas de las funcionalidades del protocolo actual. La posterior necesidad de un protocolo de transporte no fiable y sin control de flujo derivó en la separación de IP de TCP y la generación de UDP. El 1 de enero de 1983 se produjo el lanzamiento oficial de TCP/IP que como indica el RFC 1180[6], publicado más adelante, se estructura lógicamente en capas:

- Aplicación: software que hace uso de la interconexión entre equipos
- Transporte: protocolos para el transporte fiable y el control de flujo
- Red: protocolo para el encaminamiento
- Enlace: Gestión del medio físico.

Ante las posibles pérdidas de paquetes en enlaces problemáticos el RFC 3366[7] muestra distintos protocolos de retransmisión, estos protocolos son denominados ARQ (Automatic Repeat reQuest). El objetivo de esta recomendación es su uso en la capa de enlace ya que permiten la detección de la pérdida de un paquete más rápido que TCP, siendo el ámbito de operación de estos a nivel de enlace y no extremo a extremo como ocurre con la capa de transporte. Cabe destacar que al tratarse de protocolos de control de flujo y transporte fiable es posible utilizarlos a nivel de transporte, pero son menos eficientes que el propio TCP y esto se haría solo de manera académica.

## 5.2 Situación de los entornos de desarrollo

Cómo ya se ha indicado en la introducción, este proyecto se desarrollará en dos entornos distintos, por lo que se analizará la situación de cada uno de cara a la problemática por separado.

- Mathematica
- Omnet++

### 5.2.1 Mathematica

En el caso de Mathematica se deberán generar librerías para la puesta en práctica de la teoría de colas y para simular la transmisión de diversos protocolos ARQ.

En el caso de la teoría de colas se dispone de un grupo de funciones que permite su gestión. Por el contrario, el uso de las mismas y la explotación de los resultados no se adecúan a las necesidades expuestas en los objetivos del proyecto.

Por otro lado, los protocolos ARQ no están incluidos en las librerías del entorno de desarrollo ni se ha encontrado ningún proyecto libre que realice estas simulaciones. Ante esta situación se puede afirmar que no existe este tipo de librería al alcance de la universidad.

### 5.2.2 Omnet++

El entorno de desarrollo de Omnet++ y su documentación incluyen diversos manuales y tutoriales que permiten al usuario introducirse en las simulaciones. En este sentido destaca la librería INET que incluyen los diversos protocolos utilizados en internet. Como se ha indicado anteriormente las transmisiones que se quieren emular y estudiar son protocolos de estudio y no protocolos de uso actual. Ante esta situación no se encuentran elementos de Omnet++ al alcance de la universidad que cubran las necesidades de la asignatura.

# 6 OBJETIVO Y ALCANCE

En esta sección se definirá el objetivo del proyecto así como los objetivos secundarios que completan el general.

## 6.1 Objetivo del proyecto

El objetivo de este proyecto es la realización de herramientas sencillas que permitan el estudio de las problemáticas básicas de transmisión en los diversos entornos de desarrollo utilizados en la asignatura "Rendimiento en Redes de Telecomunicación". Los elementos desarrollados deberán acelerar el proceso de aprendizaje de estos entornos para facilitar el uso de los mismos al alumnado.

## 6.2 Objetivos secundarios

En este apartado se tratarán las diversas problemáticas para las que se deben desarrollar herramientas para que el alumno realice comprobaciones. Dentro de las especificaciones queda marcado cuáles de ellas deben desarrollarse para su uso en Mathematica y cuales para el uso en Omnet++.

A continuación se enumeran los temas y se indica en qué plataforma se trabajarán.

- Teoría de colas: Mathematica
- Transmisión con protocolos ARQ: Mathematica y Omnet++
- Redes Jacksonianas: Omnet++
- Rutado fijo: Omnet++
- Multiplexación estadístico: Omnet++´

Todas las simulaciones se harán únicamente con tráfico de datos. Se asume que las negociaciones de los protocolos se han realizado con anterioridad al comienzo de la simulación. A los datos se les puede añadir cabeceras de los diversos protocolos utilizados, estos protocolos en ningún caso generarán paquetes de negociación por sí mismo. Sí que se admiten los paquetes de control ACK y NACK de los protocolos ARQ ya que son necesarios para la transmisión de datos.

### 6.2.1 Teoría de colas

La librería diseñada debe permitir al usuario modelar colas G/G/1 y G/G/2 y analizar su rendimiento. El análisis se basará en la comparación de los estadísticos obtenidos con los estadísticos teóricos. El análisis teórico deberá ser desarrollado por el alumno, mientras que los resultados del modelado podrán ser obtenidos con las funciones ofrecidas por la librería. Los estadísticos a analizar son los siguientes:

- Probabilidad de estado
- Tiempo medio de estado en el sistema
- Throughput

La librería se desarrollará para trabajar en el entorno de desarrollo Mathematica.

### 6.2.2 Transmisión con protocolos ARQ

La librería diseñada deberá permitir al alumno simular la trasmisión de paquetes con distintas características mediante los protocolos ARQ Stop & Wait y Go Back N. Los resultados obtenidos se deberán comparar con los resultados teóricos y con el modelo de colas equivalentes. El análisis de resultados será desarrollado por el alumno mientras las librerías deberán permitir la simulación de los protocolos indicados.

Para ambos protocolos, se considera que los ACK y NACK no pueden ser erróneos. En caso del protocolo Go Back N, la ventana será infinita.

Los paquetes generados por esta librería deben ser compatible con la librería "drawTX.m" utilizada en la asignatura para poder visualizar la transmisión.

Las librerías entregadas para el análisis de estos protocolos deberán desarrollarse para los entornos de desarrollo Mathematica y Omnet++.

### 6.2.3 Redes Jacksonianas

Las librerías entregadas deberán permitir la simulación de redes Jacksonianas y la comparación de los resultados con los resultados teóricos. Al igual que en los apartados anteriores, el alumno deberá desarrollar el análisis teórico mientras que la librería deberá ofrecer todas las herramientas para su simulación y extracción de resultados.

Las redes Jacksonianas se simularán en el entorno de desarrollo Omnet++.

### 6.2.4 Rutado fijo

Las herramientas entregadas deberán permitir la simulación de redes de rutado fijo y analizar los resultados obtenidos.

Las redes con rutas fijas se simularán en el entorno de desarrollo Omnet++.

### 6.2.5 Multiplexación estadística

Las librerías diseñadas deberán permitir la realización de multiplexación estadística para el análisis del efecto del mismo. Junto a la multiplexación se deberá entregar también la capacidad para demultiplexar lo multiplexado.

La simulación de la multiplexación se realizará en el entorno de desarrollo Omnet++

# 7 BENEFICIOS

Antes de analizar los beneficios aportados por el proyecto se debe tener en cuenta que, como se ha expuesto hasta el momento, este está orientado y justificado por las necesidades de la asignatura, por lo que no está orientado a la obtención de beneficios directos, sino a la realización de un trabajo concreto debido a una necesidad.

Se analizarán los beneficios desde tres puntos de vista independientes

- Beneficios técnicos
- Beneficios económicos
- Beneficios sociales

## 7.1 Beneficios técnicos

Ya se ha mencionado anteriormente que este proyecto está orientado a apoyar las herramientas de software utilizadas en la asignatura RRT, debido a esto no está dirigido a obtener un avance tecnológico directo en ningún campo concreto.

En el estado del arte se ha comprobado que los recursos disponibles en los entornos de desarrollo utilizados en la asignatura no se adecuan a las necesidades de la misma. La posibilidad de incluir dichas funcionalidades en las herramientas se considera un avance técnico.

## 7.2 Beneficios económicos

Cómo se ha indicado en la introducción de este apartado el objetivo del presente proyecto no es obtener beneficios económicos directos. El resultado del proyecto será distribuido entre los alumnos. En caso de ser distribuido se distribuirá bajo la Licencia Pública General de GNU.

Con el objetivo de obtener cierto beneficio económico del trabajo realizado existe la posibilidad de estructurar un servicio de donaciones, método habitual utilizado en el software libre.

Otro de los métodos comunes de financiación de los proyectos de software libre se basa en la personalización bajo solicitud. Se considera que un correcto diseño de las funcionalidades requeridas para Omnet++ abriría una amplia posibilidad de módulos personalizados según necesidades concretas.

## 7.3 Beneficios sociales

Los beneficios sociales del proyecto vienen derivados de la situación del mismo dentro de una asignatura. El desarrollo del mismo permitirá a los alumnos desarrollar unas prácticas acordes al avance teórico. Como se ha indicado en los beneficios anteriores, estas librerías suponen un añadido a las capacidades de los diversos entornos de desarrollo y la distribución de las mismas según la Licencia Pública General de GNU permite el uso de las mismas a quien lo requiera.

# 8 ANALISIS DE ALTERNATIVAS

En el momento de diseñar las librerías necesarias existen ciertos aspectos para los que se consideraron diversas alternativas y sus características con el objetivo de optimizar el resultado. En este apartado se analizarán y expondrán dichas problemáticas y las distintas opciones que se valoraron para solventarlas, los criterios de selección utilizados y la valoración que se dio a cada una de las soluciones para seleccionar la más óptima.

Antes de acometer este análisis se considera necesario indicar la distribución de las librerías realizada para cubrir las especificaciones. Aunque esto supone una decisión de diseño no se vio necesario el análisis de alternativas ya que la decisión se consideró clara. En este caso, se realizarán dos librerías de Mathematica, una para el tratamiento de las colas G/G/X y otra para la transmisión de protocolos ARQ. Por otro lado, debido a la reutilización de funcionalidades, todos los requisitos de funcionalidades en Omnet++ se llevarán a cabo con una sola librería. Cabe destacar también que la librería de Omnet++ estará estructurada en capas.

Los aspectos a analizar serán los siguientes:

- Diferenciación de entidades dentro de las capas de Omnet++
- Tecnología para la obtención de configuraciones complejas

## 8.1 Diferenciación de entidades dentro de las capas Omnet++

Para poder completar los distintos requisitos será necesario diseñar diversas funcionalidades en algunas de las capas. En caso de tener que realizar varios protocolos para una capa se debe valorar los beneficios y desventajas de hacerlo con un solo módulo que se configure para cada protocolo o con distintos módulos.

### 8.1.1 Alternativas

Como se ha indicado anteriormente, se ha valorado las posibilidades que ofrece realizar un único módulo por cada capa y que se elija el protocolo mediante configuración o las ofrecidas por la creación de un módulo por cada funcionalidad.

- Un módulo por capa con diferentes configuraciones (config)
- Un módulo por cada funcionalidad (protocol)

#### 8.1.1.1 Un módulo por capa con diferentes configuraciones (config)

En caso de llevar a cabo esta opción se deberá diseñar un elemento independiente por cada capa. El módulo generado deberá ser capaz de ejecutar todos los protocolos requeridos para dicha capa. Para poder seleccionar que protocolo utilizar y en qué

condiciones será necesario diseñar un método de configuración para los diversos módulos.

Esta opción disminuye la cantidad de módulos diseñados, pero por el contrario la complejidad de los mismos será mayor que en la alternativa. El hecho de modificar el funcionamiento mediante configuración, sin necesidad de modificar la estructura simplifica el uso de la librería a entregar.

#### 8.1.1.2 Un módulo por cada funcionalidad (protocol)

En este caso se deberá diseñar un elemento distinto por cada protocolo necesario para cumplir las especificaciones. Cada módulo ejecutará un único protocolo, por lo que habrá que diseñar la configuración de cada uno de ellos de manera separada.

Esta opción simplifica la configuración ya que será específica para cada protocolo, pero supone mayor tiempo de desarrollo ya que se disminuye la reutilización de código. De cara al usuario, se considera más complicado de utilizar ya que es necesario crear un escenario distinto para cada configuración, por el contrario, la extracción de estadísticas será más específica.

### 8.1.2 Criterios de selección

En este apartado se expondrán los criterios que se emplearán para la selección de la solución y el peso de cada uno de ellos.

#### 8.1.2.1 Facilidad de uso (45%)

El objetivo principal de este proyecto es la simplificar el uso de las herramientas para analizar los resultados. Teniendo en cuenta esta meta el uso de las librerías diseñadas debe ser sencillo. No es aceptable que el aprendizaje del uso de las librerías sea tan complejo como el aprendizaje del uso de las herramientas. Por esta razón, se considera uno punto importante.

Cómo se ha indicado en la descripción de cada una de las alternativas, la primera opción ofrece un uso más sencillo para el usuario. De todas formas, la diferencia no se considera grande ya que la configuración de cada protocolo será más compleja que en caso de tener configuraciones específicas.

#### 8.1.2.2 Optimización de código (20%)

En caso de realizar simulaciones de redes complejas la cantidad de entidades que deben cargar y gestionar aumenta. Ante esta situación se debe tener en cuenta la carga computacional que supone cada módulo.

En este criterio es la segunda opción, un módulo por cada protocolo, la que obtiene ventaja. Al realizar un diseño específico para un protocolo se puede optimizar la ejecución para el mismo, por el contrario, si se debe comprobar las diferentes configuraciones durante la ejecución se crea una carga extra a la ejecución que es independiente del protocolo.

#### 8.1.2.3 Extracción de estadísticas (35%)

Como se ha mencionado anteriormente, el objetivo del proyecto es que el usuario pueda realizar simulaciones y analizar sus resultados sin necesidad de conocer la herramienta al detalle. Ante esta propuesta, la facilidad para la extracción de estadísticas y su análisis se considera un punto importante.

Al igual que el criterio anterior, la especialización de los módulos diseñados para cada protocolo permite realizar una extracción de datos más concreta para cada protocolo. Debido al funcionamiento de Omnet++ mediante suscripción a señales, esta diferencia no es muy destacable.

### 8.1.3 Selección de la solución

En la siguiente tabla se evaluará cada alternativa según los criterios indicados

| Criterios | Ponderación | Config | Protocol |
|---|---|---|---|
| Facilidad de Uso | 45 | 40 | 30 |
| Optimización de código | 20 | 10 | 20 |
| Extracción de estadísticas | 35 | 30 | 35 |
| **TOTAL** | **100** | **80** | **85** |

**Tabla 1 - Selección: diferenciación entre entidades de la misma capa**

Tras el análisis de los criterios seleccionados y el peso dado a los mismos, se decidió realizar un módulo distinto para cada uno de los protocolos que se consideren necesarios para el cumplimiento de las especificaciones. A pesar de no destacar en la característica más importante, la especialización de cada módulo otorga un margen de maniobrabilidad mayor en el resto de aspectos a tener en cuenta.

## 8.2 Tecnología para la obtención de configuraciones complejas

Tras decidir que se diseñará un módulo distinto para cada protocolo necesario, la configuración se simplifica considerablemente. Por el contrario, siguen existiendo algunos módulos que necesitarán la recepción de parámetros complejos o estructurados, por

ejemplo los multiplexores, demultiplexores y routers. Omnet++ posee la capacidad para introducir parámetros de forma sencilla, se considera que esta capacidad solventa la introducción simple de parámetros para el resto de módulos, pero en el caso de las configuraciones estructuradas se estima necesaria una valoración más específica. Cabe destacar que la funcionalidad básica será utilizada en todo caso ya que siempre hará falta un parámetro de referencia que indique donde encontrar la configuración estructurada.

### 8.2.1 Alternativas tecnológicas

Debido a la naturaleza de los experimentos, simulación de la transferencia de datos y no de la negociación de los protocolos, las configuraciones tiene que estar disponibles al comienzo de la prueba ya que será el momento en el que se carguen. Esta limitación nos ofrece dos opciones, hacer uso de la funcionalidad de parámetros de Omnet++ o la generación de ficheros de configuración. Se considera que los ficheros pueden ser tanto de texto plano como en formato XML. Dentro de los ficheros de texto se valoran distintas interpretaciones de los mismos.

- Parámetros
- Ficheros de texto plano
  - Formato
  - Palabras clave
- Ficheros XML

#### 8.2.1.1 Parámetros

La funcionalidad de Omnet++ nos permite introducir parámetros a los módulos. Estos parámetros pueden ser números, cadenas de caracteres, distribuciones aleatorios, arrays, etc. En caso de querer realizar configuraciones complejas con esta funcionalidad se considera que sería necesario estructurar strings o arrays.

Esta opción se considera sencilla para el usuario, pero su capacidad de estructurar datos y recibir una gran cantidad de los mismos es limitada.

#### 8.2.1.2 Ficheros de texto plano

Como ya se ha indicado anteriormente, para obtener datos mediante esta tecnología se debería definir un formato a seguir por los ficheros. Se valoró si dicho formato asignará un valor a cada línea o si será el usuario quien decida el orden de introducción de los parámetros utilizando palabras clave para identificar que parámetro se está introduciendo.

##### *8.2.1.2.1 Formato*

En este caso se debería definir un formato para los ficheros y los archivos generados para configurar deberán seguir el mismo. Este formato por líneas limita en parte la capacidad de ramificar los datos ya que queda preestablecido la cantidad de elementos hijos que puede tener un elemento padre. Por otro lado, se debe destacar que se simplifica la introducción de datos.

#### *8.2.1.2.2 Palabras clave*

En este caso se deberían definir palabras clave para utilizar al comienzo de cada línea. Las palabras serían utilizadas como etiquetas que identificarían los datos en esa línea y las etiquetas esperadas a continuación.

Esta opción permite mayor estructuración y versatilidad que la otra opción ofrecida por los ficheros de texto. Por el contrario, la introducción de datos se considera algo más complicada debido a la necesidad de conocer la estructura y palabra utilizadas en el diseño de los ficheros.

### 8.2.1.3 XML

Para la recogida de datos mediante este tipo de ficheros, se debería definir el árbol de etiquetas correspondientes a la estructura de los datos así como los atributos y características de las mismas. Esto se debería realizar de tal manera que cada tag tenga un propósito único que el usuario debería conocer.

Esta tecnología es una gran herramienta de estructuración y ofrece la capacidad de formar arboles de datos complejos. Por el contrario, se considera menos conocida y por tanto el usuario necesitaría más tiempo para adaptarse a ella.

## 8.2.2 Criterios de selección

En este apartado se expondrán los criterios para la selección de la solución y el valor que se asignará a los mismos así como la razón de dicha ponderación.

### 8.2.2.1 Estructuración (35%)

Como se ha indicado en la introducción de esta problemática, el objetivo no es la inserción de datos sencillos, sino dar solución a la obtención de configuraciones complejas. Para poder insertar y recibir este tipo de datos es necesario describir una estructura que acoja los distintos valores. Ante esta necesidad, se considera importante la capacidad de estructuración de las distintas alternativas valoradas

Esta capacidad es la ventaja principal de los ficheros XML, pero los ficheros de texto con palabras clave también ofrecen la capacidad de estructurar los datos.

#### 8.2.2.2 Introducción de datos (35%)

Se ha repetido en multitud de ocasiones que el objetivo principal del proyecto es la simplificación del uso de las herramientas necesarias para los alumnos que deben analizar el rendimiento de redes. En este sentido, se considera importante la facilidad para introducir datos.

En este sentido, la funcionalidad de parámetros de Omnet++ se considera la más sencilla, mientras que no se observa mayor complicación en un fichero con formato. Por el contrario, la necesidad de palabras clave en un fichero de texto o las etiquetas y atributos de un fichero XML sí que imponen cierta dificultad y necesidad de aprendizaje.

#### 8.2.2.3 Versatilidad (30%)

La tecnología utilizada para obtener las configuraciones complejas será utilizada en varios módulos, por lo que será necesario que la tecnología seleccionada sea capaz de dar solución a todas ellas.

En este caso es la funcionalidad de parámetros de Omnet++ la que se ve más afectada ya que ofrece opciones limitadas mientras que el resto de opciones se pueden ajustar más a las necesidades que puedan surgir.

### 8.2.3 Selección de la solución

En la siguiente tabla se evalúa cada alternativa según los criterios indicados anteriormente.

| Criterios | Ponderación | Parámetros | Fichero de texto | | XML |
|---|---|---|---|---|---|
| | | | Formato | Palabra calve | |
| Estructuración | 35 | 5 | 15 | 35 | 35 |
| Introducción de datos | 35 | 35 | 30 | 10 | 15 |
| Versatilidad | 30 | 10 | 25 | 30 | 30 |
| **TOTAL** | **100** | **50** | **70** | **75** | **80** |

**Tabla 2 - Selección: tecnología para la obtención de configuraciones**

Aunque la opción de los ficheros XML no destaca en solitario, se considera que las introducción de datos con esta tecnología es más sencilla que con ficheros definidos mediante palabras clave.

# 9 ANALISIS DE RIESGOS

La realización de todo proyecto lleva consigo una serie de riesgos que pueden afectar a la realización del mismo con distinto impacto. Ante esta situación se debe realizar un análisis de los posibles problemas que puedan surgir a lo largo del proyecto y los planes de contingencia o de reacción según el tipo de riesgo.

A continuación se enumeran los riegos valorados, la tabla con la probabilidad de que ocurra cada uno de ellos y el impacto que tendría en el proyecto en caso de ocurrir.

1. Perdida del código
2. Actualización de los entornos de desarrollo
3. Baja de recursos humanos
4. Inoperatividad de recursos informáticos

La tabla 4 muestra la probabilidad y el impacto que cada uno de ellos tendría en el proyecto si no se pone solución. A continuación se analizará cada uno de ellos incluyendo la medida el plan de contingencia o reacción según corresponda. Esta es la nomenclatura utilizada:

| **ESCALA** | | |
|---|---|---|
| impacto | probabilidad | riesgo |
| **MA**: muy alto | **MA**: prácticamente seguro | crítico |
| **A**: alto | **A**: probable | importante |
| **M**: medio | **M**: posible | apreciable |
| **B**: bajo | **B**: poco probable | bajo |
| **MB**: muy bajo | **MB**: muy raro | despreciable |

**Tabla 3 - escala de análisis de riesgos**

| **RIESGO** | | **probabilidad** | | | | |
|---|---|---|---|---|---|---|
| | | MB | B | M | A | MA |
| **impacto** | MA | 4 | 1 | | | |
| | A | | | | | |
| | M | | 3 | | | |
| | B | | | 2 | | |
| | MB | | | | | |

**Tabla 4 - distribución de riesgos**

## 9.1 Pérdida del código

Puede ocurrir que por diversas razones el código ya generado se pierda o se corrompa de tal manera que no se pueda acceder al mismo. Esto supondría la pérdida del desarrollo realizado hasta el momento.

En este caso, se establecerá un plan de contingencia para minimizar el impacto de este problema en el proyecto. Se establecerá el uso de un sistema de control de versiones distribuido. Esta solución permite tener un backup de del código además de permitir gestionar las diferentes versiones que se desarrollan del mismo.

## 9.2 Actualización de los entornos de desarrollo

Es posible en el transcurso del proyecto que alguno de los entornos de desarrollo, Mathematica y Omnet++, para los que es necesario generar librerías sean actualizados. Estas renovaciones podrían suponer la deprecación de funciones utilizadas o la inclusión de nuevas funciones que ofrezcan una mejor solución para el desarrollo.

Las actualizaciones suelen anunciarse con antelación, por esta razón el plan de contingencia se basa en la comprobación periódica de la situación de los entornos de desarrollo.

El plan de acción se activará en el momento de conocer el lanzamiento de una nueva versión. El primer paso es analizar los cambios que la actualización supone y valorar si pueden afectar el proyecto. En caso de que estos cambios afecten al proyecto se deberán realizar los cambios correspondientes en el diseño y replanificar desde el estado del proyecto junto con el nuevo diseño.

## 9.3 Baja de recursos humanos

Este proyecto está realizado por pocas personas, por lo que la incapacidad de alguno de ellos supone un gran impacto en los plazos del proyecto.

Si durante un periodo el responsable de alguna de las tareas no pueda realizarla en plazo debido a una incapacidad se tratará de realizar horas extra a posteriori y se replanificará el proyecto en caso de que el retraso sea irrecuperable.

## 9.4 Inoperatividad de recursos informáticos

Al tratarse de un proyecto software todo su desarrollo y documentación se realiza mediante equipos informáticos, la incapacidad de operar de los mismo puede paralizar por completo el proyecto. Debido a esto se considera de alto impacto, aunque se algo con poca probabilidad de ocurrencia.

En este caso no se dispone de un plan de contingencia más que las medidas habituales de mantenimiento de este tipo de equipos. El plan de acción se basa en la adquisición de un nuevo ordenador con capacidades suficientes para utilizar los entornos de desarrollo y el gestor de versiones utilizados

# 10 DISEÑO DE ALTO NIVEL

En el presente apartado se analizará la solución final de diseño derivada del análisis de alternativas anteriormente expuesto. Este diseño propone la manera más óptima de cumplir todas las especificaciones recibidas.

Cómo ya se ha indicado en la introducción del análisis de alternativas se decide realizar en total 3 librerías, dos para el entorno de desarrollo Mathematica, una para las colas G/G/X y otra para la transmisión de protocolos ARQ, por otro lado, se realizará una para Omnet++ que deberá dar solución a todas las necesidades en este entorno de desarrollo.

Se orientarán las librerías entorno a estructuras comunes. Todos las funciones o módulos de cada una de las librerías aceptarán como entrada o generarán como salida un tipo concreto de estructura. Esta capacidad permitirá que las bibliotecas no sean cerradas y permitan una ampliación simple de las capacidades de las mismas.

## 10.1 Librería de Mathematica para colas G/G/X

Esta librería debe permitir modelar colas del tipo G/G/X, con lo que la distribución de tiempo entre paquetes y tamaño, o tiempo de procesado, de los mismos puede seguir cualquier distribución aleatoria. Las colas serán de 1 o 2 servidores.

El primera paso es decidir que debe haber una estructura que represente al paquete. Mathematica no permite la generación de clases o estructuras como tal, pero sí que trabaja con arrays numéricos y matrices, que podemos considerar listas de arrays. Antes esta situación se definirá un tipo de array que defina un paquete y una lista de estos arrays representará una lista de paquetes que ha podido o no pasar ya por la cola. La definición de los campos de estos array se realizará en el diseño de bajo nivel.

La librería se divide en dos partes, por un lado, la modulación de colas y por otro la obtención de estadísticos

### 10.1.1 Modulación de colas

La ilustración 1 muestra el diagrama general que se quiere seguir para modelar las colas requeridas. Las colas G/G/X son independientes del tipo de entrada, por lo que se diseñará un único módulo que genere la lista de paquetes que deberá ser introducida en la cola.

Como se muestra en la imagen se diseñarán las colas indicadas G/G/1 y G/G/2, pero se considera interesante también el diseño de una cola de un único servidor con cola finita. Todas las colas recibirán el mismo tipo de entrada.

Por otro lado, se considera necesaria la capacidad de que con una sola llamada se simule completamente cada una de las colas, esto es, sin necesidad de conocer la comunicación entre la generación de entradas y la cola. Esto es representado por la flecha amarilla que envuelve todos los módulos. Esta llamada deberá tener una entrada similar al generador de paquetes de entrada y una salida idéntica a cada uno de las colas.

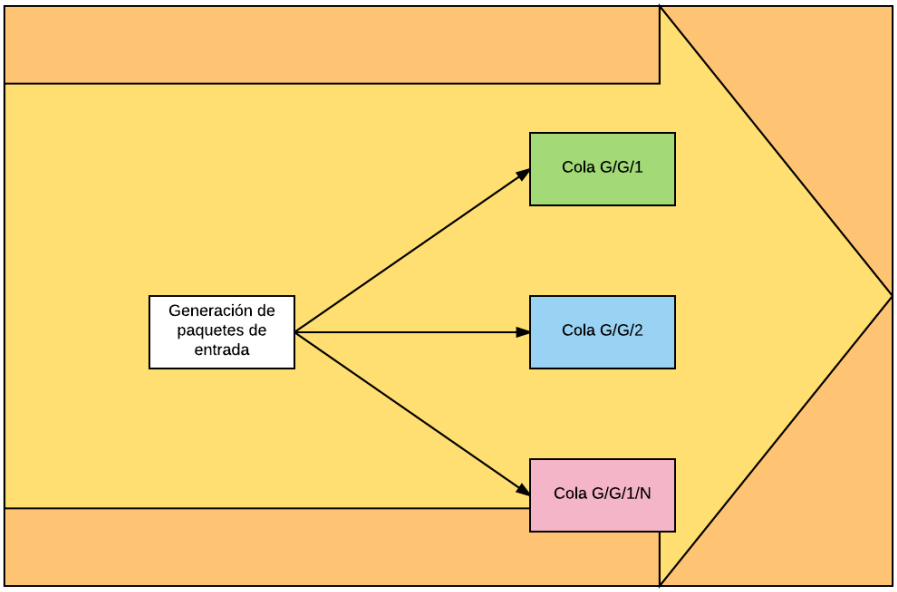


**Ilustración 1 - Estructura general del modelado de colas**

### 10.1.2 Obtención de estadísticos

Se deberán diseñar funciones que permitan al usuario obtener los siguientes datos sin necesidad de conocer la estructura de la lista de paquetes extraída de las colas:

- Estado del sistema
- Probabilidad de estado del sistema
- Tiempo medio en el sistema
- Throughput

## 10.2 Librería de Mathematica de protocolos ARQ

Esta librería debe ser compatible con la librería "drawTX.m", esta compatibilidad se basa en tener una misma interpretación de los paquetes. En este caso, se deben diseñar módulos de transmisión Stop & Wait y Go Back N. Al igual que en el caso de la librería diseñada para las colas está tendrá un único generador de paquetes y un procesado independiente para cada una de las colas.

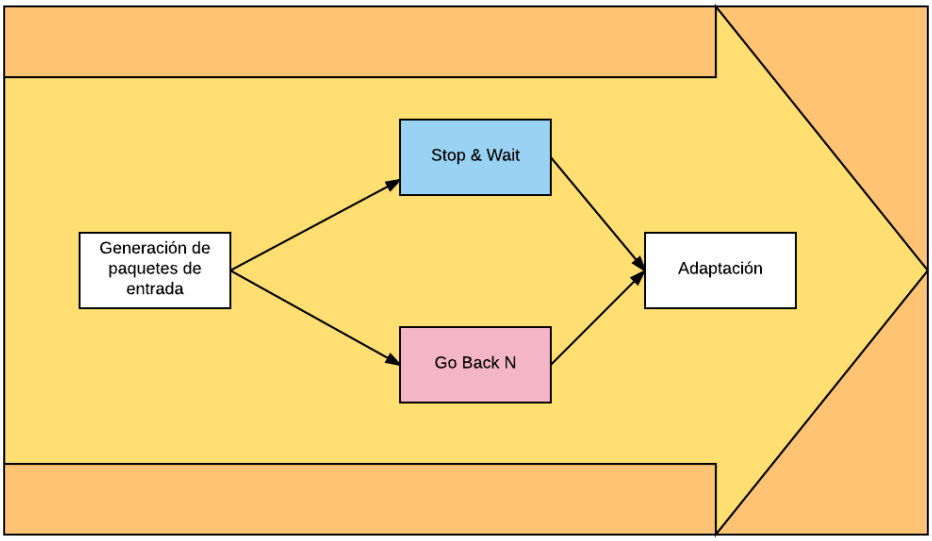


**Ilustración 2 - Estructura general de la simulación ARQ**

Con el objetivo de que la simulación de la transmisión sea lo más acorde posible a la realidad, la simulación se hará en función al tamaño de paquete y a la velocidad de la línea. Esto no es acorde a la interpretación de la librería "drawTX.m", por lo que es necesaria una adaptación.

Al igual que en otra librería, se considera necesaria la capacidad de simular completamente una transmisión sin conocer el funcionamiento completo del módulo.

## 10.3 Librería Omnet++

Cómo se ha indicado en las especificaciones esta librería debe dar solución a multitud de problemáticas:

- Transmisión con protocolos ARQ
  - Compatible con la simulación en Mathematica
  - Genérico
- Redes Jacksoniano
- Rutado fijo
- Multiplexación estadístico

Debido a la variedad de situaciones a tratar y la relación entre ellas se decide un diseño en capas que permita reutilizar módulos y extender las posibles simulaciones. La distribución en capas se realizará basándose en el RFC 118[6], modelo TCP/IP. Por el contrario, para la comunicación en de las mismas y el direccionamiento se ha utilizado la recomendación ISO/IEC 7498-1[8].

### 10.3.1 Distribución de capas

A continuación se muestra la lista de las capas y su funcionalidad, basada en el RFC 118 como se ha indicado anteriormente:

- **Aplicación:** será la capa encargada de la generación y recepción de tráfico
- **Transporte:** capa encargada del control de flujo
- **Red:** nivel encargado del encaminamiento
- **Enlace:** capa que gestiona la conexión física

Como se puede observar en la ilustración 3 se diseñará una única interfaz común a todas las capas, Inter Layer, esta unión de la interfaz permite que una misma capa pueda comunicarse con distintos niveles inferiores sin necesidad de modificaciones. La imagen nos muestra las distintas combinaciones de capas que se pueden realizar, teniendo en cuenta que las capas superiores se pueden eliminar en elementos intermedios de la red.

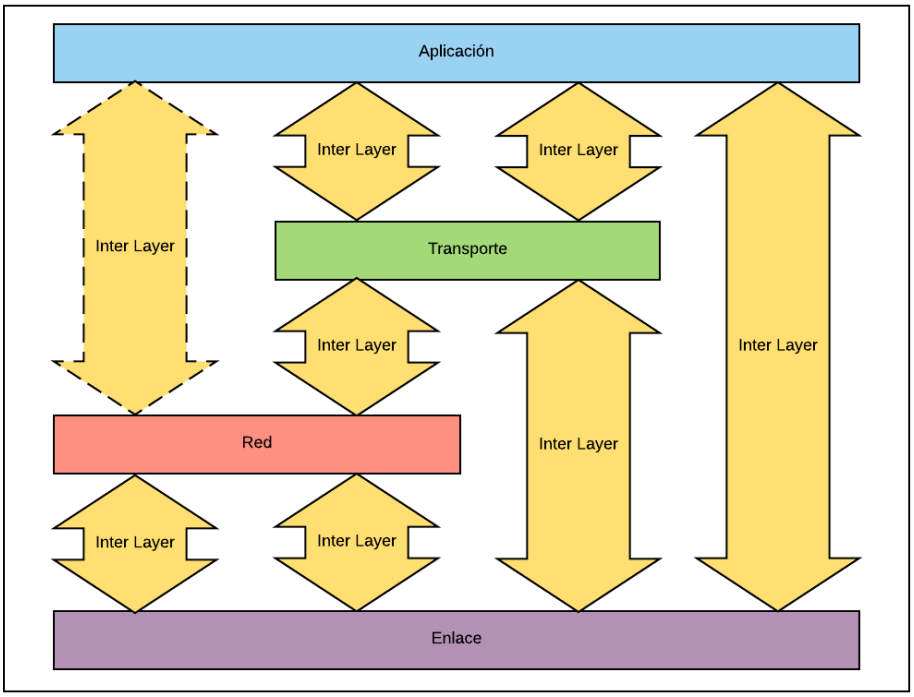


**Ilustración 3 - Estructura de librería de Omnet++**

Para los elementos finales siempre son necesarias las capas de aplicación y enlace, mientras que se podrían realizar elementos intermedios, como routers que no necesiten el nivel de aplicación. La capa de transporte al encargase del control de flujo y ser un capa extremo-extremo tiene como objetivo su uso en equipos finales.

Se diseñará un paquete distinto para cada una de las capas. Este paquete debe ser capaz de encapsular cualquiera de las capas superiores.

Como se ha indicado en el análisis de alternativas, se considera que la generación de un módulo por cada protocolo o funcionalidad específica de cada capa. A continuación se mostrarán los distintos módulos a diseñar de cada capa.

#### 10.3.1.1 Aplicación

Esta capa es la encargada de la generación y recogida del tráfico de datos. Se generará un módulo distinto para cada una de estas funciones:

- **Injector:** módulo encargado de la generación del tráfico
- **Dump:** módulo encargado de la recogida del tráfico

La imagen a continuación muestra cómo el tráfico viajará siempre desde el módulo injector hasta el módulo de recogida. Si se quiere simular la comunicación entre dos extremos de la red se deberá incluir un módulo de cada tipo en cada uno de los equipos.

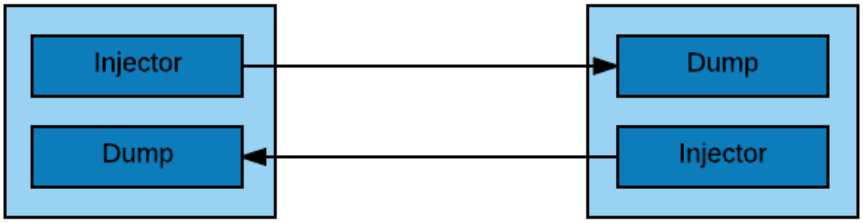


**Ilustración 4 - Comunicación de módulos de aplicación**

#### 10.3.1.2 Transporte

Este nivel es el encargado de la gestión de flujo. Se diseñarán módulos para los dos protocolos ARQ indicados en las especificaciones: Stop & Wait y Go Back N

Ambos módulos tendrán una funcionalidad idéntica para la recepción de mensajes y generación de ACK. Ante esta situación, el protocolo de transmisión lo establece el equipo emisor, el receptor siempre tendrá la misma política de respuesta. La ilustración 9 muestra dos transmisiones de datos una desde el host1 al host2 y otra en sentido inverso. Por tanto, los protocolos serán de la siguiente manera

- Host1 → Host 2: Stop & Wait
- Host2 → Host2: Go Back N

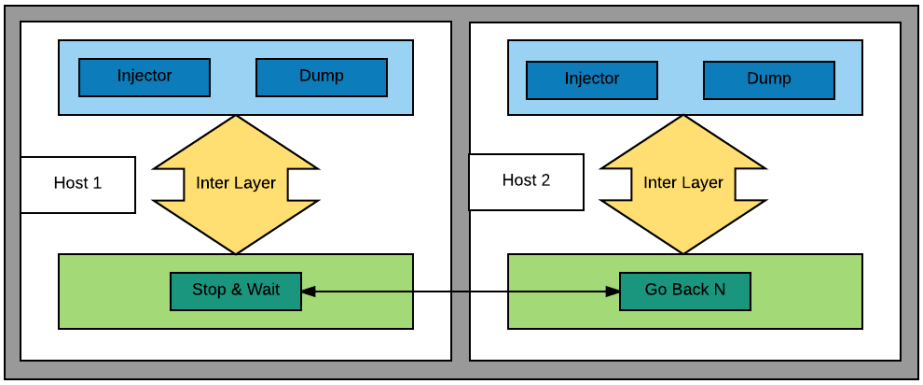


**Ilustración 5 - Comunicación de módulos de transporte**

#### 10.3.1.3 Red

Este nivel es el encargado del rutado. Aunque las especificaciones indican que se deben realizar dos tipos de rutado, se considera que el comportamiento es similar, por lo que se realizará un único módulo que sea capaz de realizar ambas funciones.

Este módulo requiere configuración compleja para las rutas. Cómo se ha identificado en el análisis de alternativas esto ser realizará mediante ficheros XML, por lo que será necesario el diseño de un fichero XML que permita realizar la configuración de las rutas. El resto de parámetros se realizará mediante la función de parámetros de Omnet++.

#### 10.3.1.4 Enlace

Esta capa tiene como objetivo gestionar el enlace, pero se ofrecerán más capacidades. Por un lado se generará módulos con el único objetivo de gestionar el enlace y por otro lado módulos que además incluyan gestión de flujo.

La conexión entre los enlaces de distintos módulos se deberá realizar mediante enlaces físico y no los enlaces lógicos utilizados en la interfaz inter layer.

- **Gestión del enlace**
  - **Fisico:** Gestiona el enlace, encapsula los paquetes y gestiona el error
  - **Simple_fisico:** este módulo no realiza ningún tipo de comprobación ni encapsula los paquetes, solo gestiona que el uso del enlace sea correcto
- **Gestión de flujo**
  - **SenderSW:** emisor con el protocolo Stop & Wait
  - **SenderGBN:** emisor con el protocolo Go Back N
  - **ReceiverACK:** recibe paquetes y contesta con el ACK o NACK correspondientes.

#### *10.3.1.4.1 Gestión del enlace*

La transmisión de información entre los módulos de esta categoría será full-duplex. Ambos módulos deberán ser capaces de gestionar el envío de paquete a través del enlace, recibir paquetes de la capa superior y redirigir los paquetes recibidos a la capa superior.

#### *10.3.1.4.2 Gestión de flujo*

En este caso, la comunicación será unidireccional. El emisor decidirá el protocolo y el receptor redirigirá los paquetes a la capa superior. Si en ambos host se dispone de un módulo emisor y otro receptor se podrá realizar una comunicación full-duplex.

Se asume que los ACK y NACK no puedes ser erróneos.

### 10.3.1.5 Multiplexores

Aunque esta no es una capa como tal, sino una funcionalidad, se diseñarán módulos específicos pare ellos, dos multiplexores y un solo demultiplexor

- Multiplexores:
  - Multiplexor simple
  - Multiplexor con prioridad
- Demultiplexor

La entrada de los multiplexores y la salida del demultiplexor harán uso de la interfaz inter layer, por lo que tendrán necesidad de una capa inferior, la capa de enlace.

Los paquetes recibidos de la capa inferior serán encapsulados en la cabecera de multiplexación. Esta cabecera indica el número de línea que identifica la trama, de tal manera que el demultiplexor pueda identificar por qué salida mandar el paquete encapsulado.

Se considera que tanto los multiplexores como el demultiplexores requieren configuraciones complejas para lo que habrá que diseñar ficheros XML que asocien las entradas salidas con las líneas y las prioridades.

La arquitectura se puede apreciar en la ilustración 6. Las flecha negras identifican enlaces físicos. La imagen muestra que la conexión entre el multiplexor y el demultiplexor se realizará mediante enlaces físicos y no lógicos.

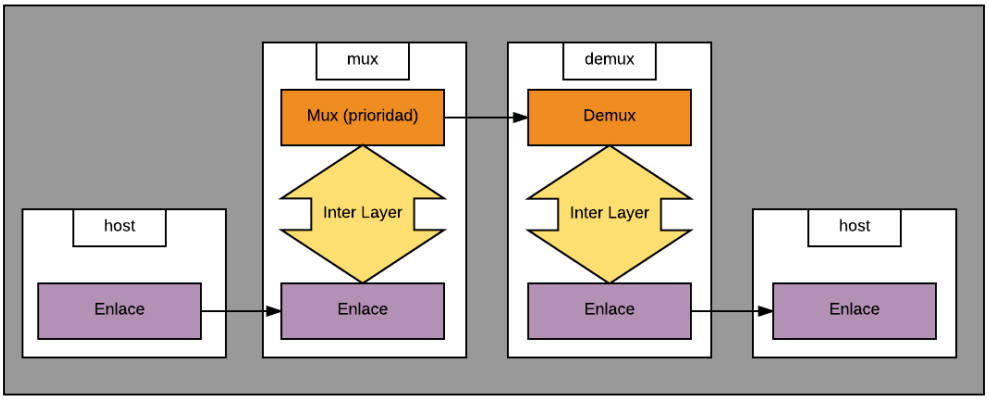


**Ilustración 6 - Estructura de módulos de multiplexación**

A continuación se analizarán las funciones de cada uno de los módulos.

#### *10.3.1.5.1 Multiplexor simple*

Los paquetes recibidos de distintas entradas deberán ser clasificados en distintas líneas. Paquetes de distintas entradas podrán estar en la misma línea. La línea será utilizada por el demultiplexor para identificar la salida.

#### *10.3.1.5.2 Multiplexor con prioridades*

Las líneas seguirán el mismo funcionamiento que en el caso del multiplexor simple. Las prioridades serán asignadas a cada entrada. Esto es, podría haber paquetes que siendo de una misma línea tengan distintas prioridades.

Cada entrada tendrá una línea y una prioridad. Las líneas identificarán la salida del demultiplexor por que las que se mandaran los paquetes y la prioridad el orden de emisión de los paquetes.

#### *10.3.1.5.3 Demultiplexor*

El módulo recibirá paquetes encapsulados. Deberá desencapsularlos y retrasmitirlos por las salidas que las líneas identifiquen. Se podrá configurar cada línea para retrasmitirse por una salida. Se podrán asociar más de una línea a cada salida.

# 11 METODOLOGÍA

En este apartado se analizarán las metodologías de desarrollo utilizadas, los recursos humanos necesarios para su realización y finalmente se analizarán los paquetes y las tareas que los componen.

## 11.1 Metodología de desarrollo

La naturaleza de las librerías necesarias para Mathematica es distinta que la de la librería de Omnet++, ante esta situación, la metodología de desarrollo utilizada para cada entorno de desarrollo diferente.

Las librerías de Mathematica son bloques de funciones cerradas que interoperan mediante una interpretación común de la estructura de datos. Ante esta situación, se considera que la metodología básica de los proyectos software, el desarrollo en 'V'. Se comenzará analizando los objetivos del proyecto para el plan de pruebas de aceptación. Tras esto se diseñará una solución general que desembocará en el plan de pruebas de integración. El último paso del diseño es la generación del diseño de bajo nivel y el plan de pruebas unitarias. Finalmente se desarrollará la solución y se verificarán los planes de prueba en orden inverso al creado.

Respecto a la librería de Omnet++ se considera que la metodología básica de los proyectos software no se ajusta a las necesidades de esta librería. En este caso, se considera que un desarrollo en espiral que aumente progresivamente las funcionalidades se ajustan mejor al tipo de desarrollo que se requiere. Se comienza realizando un desarrollo completo de una versión reducida de los objetivos, una vez realizados se establecen nuevos objetivos y se vuelve a realizar el desarrollo. Se considera que este modelo es más adecuado ya que se aplica el aprendizaje de cada fase en las fases posteriores.

## 11.2 Recursos humanos

El equipo de trabajo que se ha hecho cargo de la elaboración del proyecto está comprendido por los siguientes miembros:

- Director del proyecto: Se trata de un ingeniero senior con experiencia en la supervisión de este tipo de proyectos. Su labor es la de guiar el desarrollo del proyecto.
- Diseñador: Un ingeniero en telecomunicaciones junior con conocimientos de desarrollo de proyectos software. La labor de este miembro es la de realizar un análisis de la situación y de la problemática a cubrir y realizar un diseño que resuelva dicha problemática de la manera más eficiente.

- Desarrollador: Ingeniero en telecomunicaciones junior con conocimientos de programación en los diferentes entornos de desarrollo. Su labor es la codificación de las soluciones de diseño generada por el diseñador.
- Administrador: Su función es la de atender todas las tareas administrativas del proyecto así como realizar la documentación asociada al mismo. Se requieren conocimientos de ofimática para esta tarea.

## 11.3 Definición de los paquetes de trabajo

La planificación de todo proyecto se estructura en paquetes de trabajo que están organizados en tareas y tienen como objetivo obtener uno o más entregables. A continuación se enumerarán los paquetes de trabajo y se realizará una descripción de cada uno de ellos así como sus productos finales.

- Definición del trabajo
- Librería de Mathematica para teoría de colas
- Librería de Mathematica para protocolos ARQ
- Librería de Omnet++
- Gestión del proyecto

### 11.3.1 Definición del trabajo

Este paquete se basa en el análisis de la problemática presentada y la definición del objetivo y los objetivos secundarios del proyecto. Tras analizar las necesidades a cubrir se establecerá el entregable final del proyecto así como sus requisitos.

- Entregables:
  - Alcance
  - Plan de pruebas de aceptación

### 11.3.2 Librería de Mathematica para teoría de colas

Este bloque incluye el diseño, desarrollo y pruebas de la librería que tiene como objetivo tratar la teoría de colas. Del mismo modo se divide en estos subpaquetes.

#### 11.3.2.1 Diseño de alto nivel

Se comprobará la situación actual del entorno de desarrollo para dar solución a la problemática planteada. Se realizará el diseño general de esta librería y las condiciones que debe cumplir

- Entregables
  - Diseño de alto nivel
  - Plan de pruebas de integración

#### 11.3.2.2 Diseño de bajo nivel

Se realizará el diseño de cada módulo y los requisitos de cada uno de ellos

- Entregables
  - Diseño de bajo nivel
  - Plan de pruebas unitarias

#### 11.3.2.3 Codificación y pruebas

Dentro de este paquete se codificará cada uno de los módulos y se realizarán las pruebas unitarias de cada uno de ellos. Tras esto se realizará el ensamblado y sus pruebas correspondientes. Finalmente se comprobará que la librería cumple las pruebas de aceptación.

- Entregable:
  - Librería comprobada

### 11.3.3 Librería de Mathematica para protocolos ARQ

Este paquete incluye el diseño, desarrollo y pruebas de la librería que tiene como objetivo tratar la teoría de colas. Se divide en los mismos subpaquetes que el paquete anterior por lo que no se describirán.

### 11.3.4 Librería de Omnet++

Este bloque incluye todos los ciclos realizados para la librería de Omnet++. A continuación se indica cada una de las fases.

#### 11.3.4.1 Fase 1: trasmisión de paquetes

En esta fase se establecerá la estructura básica de comunicación entre capas y elementos básicos de comunicación. Se realizará el diseño, desarrollo y las pruebas de los módulos de generación y recepción de tráfico así como los elementos mínimos de gestión del enlace.

- Entregables:
  - Diseño de interfaz intercapas
  - Diseño de los módulos generados
  - Plan de Pruebas
  - Desarrollo de los módulos comprobados

#### 11.3.4.2 Fase 2: control de flujo

Esta fase incluye los módulos de control de flujo en el propio enlace como el control de flujo extremo a extremo.

- Entregables:
  - Diseño de los módulos generados
  - Plan de pruebas
  - Desarrollo de los módulos comprobados

#### 11.3.4.3 Fase 3: rutado

Esta fase tiene como objetivo el desarrollo de las funcionalidades de rutado dentro de las capas definidas

- Entregables:
  - Diseño de los módulos de rutado
  - Plan de pruebas
  - Desarrollo de los módulos comprobados

#### 11.3.4.4 Fase 4: multiplexación

Esta fase tiene como objetivo el desarrollo de las funcionalidades de multiplexación según la conexión intercapas diseñada.

- Entregables:
  - Diseño de los módulos de multiplexación
  - Plan de pruebas
  - Desarrollo de los módulos comprobados

#### 11.3.4.5 Arquitectura y pruebas

Este apartado tiene como objetivo generar arquitecturas de ejemplo que permitan comprobar la interoperabilidad entre los distintos módulos.

- Entregable:
  - Arquitecturas de ejemplo

### 11.3.5 Gestión del proyecto

Este paquete tiene como objetivo realizar la gestión continua del proyecto.

## 11.4 Distribución de los paquetes y tareas

A continuación se muestra las tareas que hay en cada paquete y las horas que se requiere que cada recurso de los expuestos anteriormente invierta en cada una de ellas.

| TAREA | Director de proyecto | Diseñador | Desarrollador | Administrador |
|---|---|---|---|---|
| | horas | horas | horas | horas |
| Definición del trabajo | 7 | 19 | 0 | 1 |
| Requisitos de teoría de colas | 1 | 3 | 0 | 0 |
| Requisitos de protocolos ARQ | 1 | 3 | 0 | 0 |
| Requisitos de redes Jacksonianas | 1 | 3 | 0 | 0 |
| Requisitos de rutado | 1 | 3 | 0 | 0 |
| Requisitos de Multiplexación | 1 | 3 | 0 | 0 |
| Estructuración de librerías | 1 | 2 | 0 | 0 |
| Plan de pruebas de aceptación | 1 | 2 | 0 | 1 |
| Librería de Mathematica para teoría de colas | 4 | 31 | 52 | 2 |
| Diseño de alto nivel | 4 | 12 | 0 | 1 |
| Estado del arte | 0 | 5 | 0 | 0 |
| Diseño general | 2 | 5 | 0 | 0 |
| Plan de pruebas de integración | 2 | 2 | 0 | 1 |
| Diseño de bajo nivel | 0 | 19 | 0 | 1 |
| Diseño de los funciones | 0 | 17 | 0 | 0 |
| Plan de pruebas unitarias | 0 | 2 | 0 | 1 |
| Codificación y pruebas | 0 | 0 | 52 | 0 |
| Codificación | 0 | 0 | 40 | 0 |
| Pruebas unitarias | 0 | 0 | 6 | 0 |
| Pruebas de integración | 0 | 0 | 3 | 0 |
| Pruebas de aceptación | 0 | 0 | 3 | 0 |
| Librería de Mathematica para protocolos ARQ | 4 | 29 | 50 | 2 |
| Diseño de alto nivel | 4 | 15 | 0 | 1 |
| Estado del arte | 0 | 8 | 0 | 0 |
| Diseño general | 2 | 5 | 0 | 0 |
| Plan de pruebas de integración | 2 | 2 | 0 | 1 |
| Diseño de bajo nivel | 0 | 14 | 0 | 1 |
| Diseño de los funciones | 0 | 12 | 0 | 0 |
| Plan de pruebas unitarias | 0 | 2 | 0 | 1 |
| Codificación y pruebas | 0 | 0 | 50 | 0 |
| Codificación | 0 | 0 | 40 | 0 |
| Pruebas unitarias | 0 | 0 | 5 | 0 |
| Pruebas de integración | 0 | 0 | 3 | 0 |
| Pruebas de aceptación | 0 | 0 | 2 | 0 |
| Librería de Omnet++ | 8 | 69 | 265 | 0 |
| Fase 1: trasmisión de paquetes | 4 | 18 | 58 | 0 |
| Diseño de interfaz inter capas | 2 | 10 | 0 | 0 |
| Diseño de gestión de enlace básica | 1 | 4 | 0 | 0 |
| Diseño de capa de aplicación | 1 | 4 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| Codificación | 0 | 0 | 45 | 0 |
| Pruebas | 0 | 0 | 13 | 0 |
| Fase 2: control de flujo | 2 | 10 | 50 | 0 |
| Diseño de gestión de flujo en enlace | 1 | 5 | 0 | 0 |
| Diseño de gestión de flujo extremo a extremo | 1 | 5 | 0 | 0 |
| Codificación | 0 | 0 | 32 | 0 |
| Pruebas | 0 | 0 | 18 | 0 |
| Fase 3: rutado | 1 | 18 | 59 | 0 |
| Diseño rutado | 1 | 18 | 0 | 0 |
| Codificación | 0 | 0 | 37 | 0 |
| Pruebas | 0 | 0 | 22 | 0 |
| Fase 4: multiplexación | 1 | 23 | 61 | 0 |
| Diseño de multiplexación | 1 | 23 | 0 | 0 |
| Codificación | 0 | 0 | 48 | 0 |
| Pruebas | 0 | 0 | 13 | 0 |
| Arquitecturas y pruebas | 0 | 0 | 37 | 0 |
| Generación de redes básicas | 0 | 0 | 22 | 0 |
| Pruebas | 0 | 0 | 15 | 0 |
| Gestión del proyecto | 25 | 5 | 10 | 200 |
| Reuniones de análisis | 15 | 5 | 10 | 0 |
| Documentación | 10 | 0 | 0 | 200 |
| TOTAL | 48 | 153 | 377 | 205 |

**Tabla 5 - Paquetes de trabajo y tareas**

Una vez expuesta todas las tareas a llevar a cabo y su distribución en paquetes de trabajo, se representa el diagrama de Gantt que engloba estas tareas. El siguiente gráfico muestra la distribución temporal de las mismas que han seguido una sucesión temporal y se solapan con la documentación de cada uno de ellos.

| Id | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|
| 1 | Definición del trabajo | 6 días | lun 04/01/16 | mar 12/01/16 |
| 2 | Requisitos de teoría de colas | 5 días | lun 04/01/16 | vie 08/01/16 |
| 3 | Requisitos de protocolos ARQ | 5 días | lun 04/01/16 | vie 08/01/16 |
| 4 | Requisitos de redes Jacksonianas | 5 días | lun 04/01/16 | vie 08/01/16 |
| 5 | Requisitos de rutado | 5 días | lun 04/01/16 | vie 08/01/16 |
| 6 | Requisitos de Multiplexación | 5 días | lun 04/01/16 | vie 08/01/16 |
| 7 | Estructuración de librerías | 1 día | lun 11/01/16 | lun 11/01/16 |
| 8 | Alcance | 0 días | mar 12/01/16 | mar 12/01/16 |
| 9 | Plan de pruebas de aceptación | 0 días | mar 12/01/16 | mar 12/01/16 |
| 10 | Librería de Mathematica para teoría de colas | 56 días | mar 12/01/16 | mié 30/03/16 |
| 11 | Diseño de alto nivel | 6 días | mar 12/01/16 | mar 19/01/16 |
| 12 | Estado del arte | 2 días | mar 12/01/16 | mié 13/01/16 |
| 13 | Diseño general | 3 días | jue 14/01/16 | lun 18/01/16 |
| 14 | Plan de pruebas de integración | 1 día | mar 19/01/16 | mar 19/01/16 |
| 15 | Diseño de alto nivel | 0 días | lun 18/01/16 | lun 18/01/16 |
| 16 | Plan de pruebas de integración | 0 días | mar 19/01/16 | mar 19/01/16 |
| 17 | Diseño de bajo nivel | 11 días | mié 20/01/16 | mié 03/02/16 |
| 18 | Diseño de los funciones | 10 días | mié 20/01/16 | mar 02/02/16 |
| 19 | Plan de pruebas unitarias | 1 día | mié 03/02/16 | mié 03/02/16 |
| 20 | Diseño de bajo nivel | 0 días | mar 02/02/16 | mar 02/02/16 |
| 21 | Plan de prueba unitarias | 0 días | mié 03/02/16 | mié 03/02/16 |
| 22 | Codificación y pruebas | 39 días | jue 04/02/16 | mié 30/03/16 |
| 23 | Codificación | 30 días | jue 04/02/16 | mié 16/03/16 |
| 24 | Pruebas unitarias | 5 días | jue 17/03/16 | mié 23/03/16 |
| 25 | Pruebas de integración | 3 días | jue 24/03/16 | lun 28/03/16 |
| 26 | Pruebas de aceptación | 1 día | mar 29/03/16 | mar 29/03/16 |
| 27 | Librería contrastada | 0 días | mié 30/03/16 | mié 30/03/16 |
| 28 | Librería de Mathematica para protocolos ARQ | 54 días | vie 01/04/16 | mié 15/06/16 |
| 29 | Diseño de alto nivel | 7 días | vie 01/04/16 | lun 11/04/16 |
| 30 | Estado del arte | 3 días | vie 01/04/16 | mar 05/04/16 |
| 31 | Diseño general | 3 días | mié 06/04/16 | vie 08/04/16 |
| 32 | Plan de pruebas de integración | 1 día | lun 11/04/16 | lun 11/04/16 |
| 33 | Diseño de alto nivel | 0 días | vie 08/04/16 | vie 08/04/16 |
| 34 | Plan de pruebas de integración | 0 días | lun 11/04/16 | lun 11/04/16 |
| 35 | Diseño de bajo nivel | 9 días | mar 12/04/16 | vie 22/04/16 |
| 36 | Diseño de los funciones | 8 días | mar 12/04/16 | jue 21/04/16 |
| 37 | Plan de pruebas unitarias | 1 día | vie 22/04/16 | vie 22/04/16 |
| 38 | Diseño de bajo nivel | 0 días | jue 21/04/16 | jue 21/04/16 |
| 39 | Pruebas unitarias | 0 días | vie 22/04/16 | vie 22/04/16 |
| 40 | Codificación y pruebas | 38 días | lun 25/04/16 | mié 15/06/16 |
| 41 | Codificación | 30 días | lun 25/04/16 | vie 03/06/16 |
| 42 | Pruebas unitarias | 4 días | lun 06/06/16 | jue 09/06/16 |
| 43 | Pruebas de integración | 3 días | vie 10/06/16 | mar 14/06/16 |
| 44 | Pruebas de aceptación | 1 día | mié 15/06/16 | mié 15/06/16 |
| 45 | Librería contrastada | 0 días | mié 15/06/16 | mié 15/06/16 |
| 46 | Librería de Omnet++ | 65 días | jue 16/06/16 | mié 14/09/16 |
| 47 | Fase 1 | 15 días | jue 16/06/16 | mié 06/07/16 |

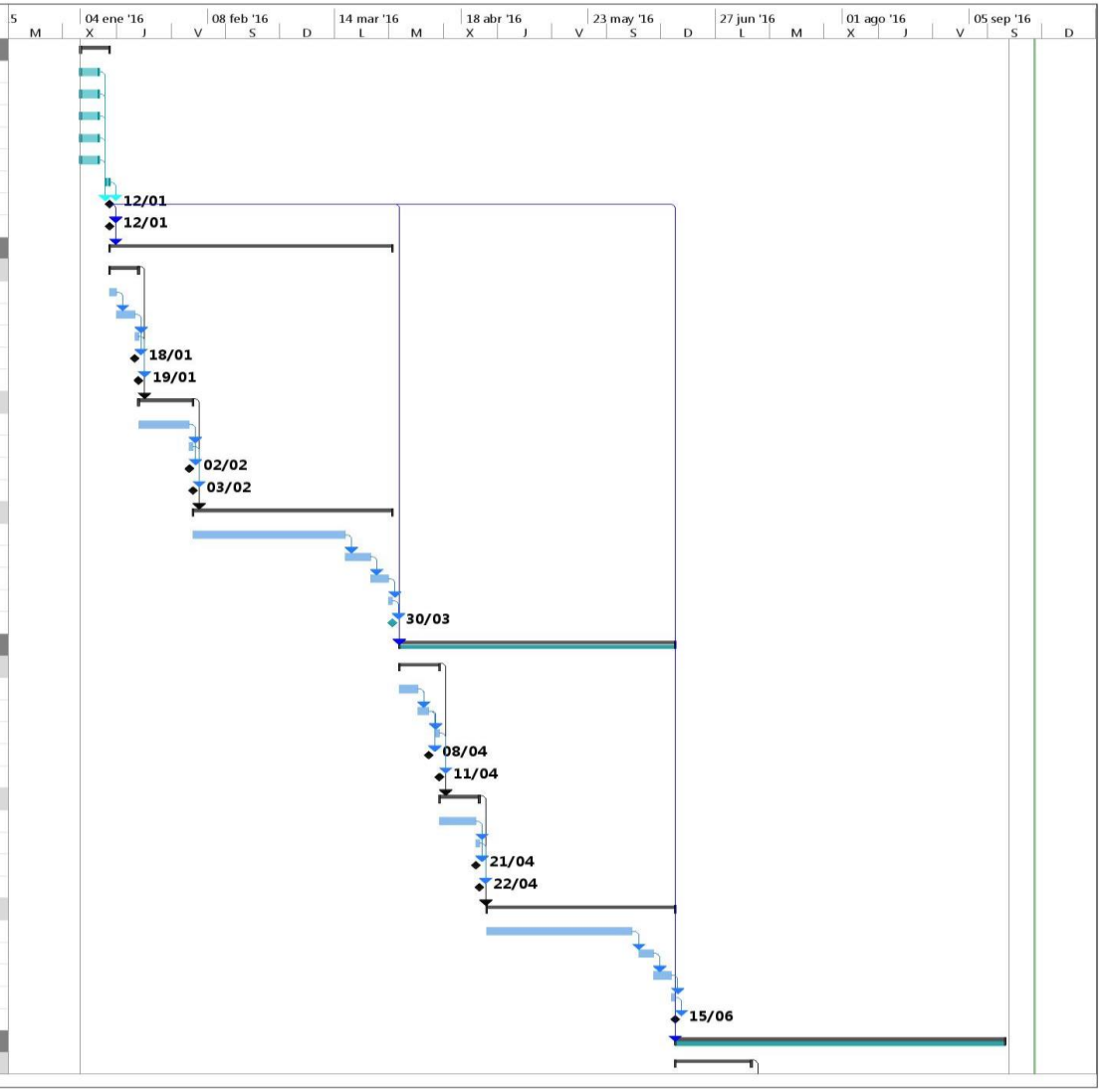


**Ilustración 7 - Diagrama gantt parte1**

| Id | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|
| 48 | Diseño de interfaz inter capas | 3 días | jue 16/06/16 | lun 20/06/16 |
| 49 | Diseño de gestión de enlace basica | 1 día | mar 21/06/16 | mar 21/06/16 |
| 50 | Diseño de capa de aplicación | 1 día | mar 21/06/16 | mar 21/06/16 |
| 51 | Codificación | 8 días | mié 22/06/16 | vie 01/07/16 |
| 52 | Pruebas | 3 días | lun 04/07/16 | mié 06/07/16 |
| 53 | Modulos y diseño | 0 días | mié 06/07/16 | mié 06/07/16 |
| 54 | Fase 2 | 11 días | jue 07/07/16 | jue 21/07/16 |
| 55 | Diseño de gestion de flujo en enlace | 2 días | jue 07/07/16 | vie 08/07/16 |
| 56 | Diseño de gestión de flujo extremo a extremo | 2 días | jue 07/07/16 | vie 08/07/16 |
| 57 | Codificación | 6 días | lun 11/07/16 | lun 18/07/16 |
| 58 | Pruebas | 3 días | mar 19/07/16 | jue 21/07/16 |
| 59 | Módulos y diseño | 0 días | jue 21/07/16 | jue 21/07/16 |
| 60 | Fase 3 | 16 días | vie 22/07/16 | vie 12/08/16 |
| 61 | Diseño rutado | 5 días | vie 22/07/16 | jue 28/07/16 |
| 62 | Codificación | 7 días | vie 29/07/16 | lun 08/08/16 |
| 63 | Pruebas | 4 días | mar 09/08/16 | vie 12/08/16 |
| 64 | Módulos y diseño | 0 días | vie 12/08/16 | vie 12/08/16 |
| 65 | Fase 4 | 18 días | lun 15/08/16 | mié 07/09/16 |
| 66 | Diseño de multiplexación | 6 días | lun 15/08/16 | lun 22/08/16 |
| 67 | Codificación | 9 días | mar 23/08/16 | vie 02/09/16 |
| 68 | Pruebas | 3 días | lun 05/09/16 | mié 07/09/16 |
| 69 | Módulos y diseño | 0 días | mié 07/09/16 | mié 07/09/16 |
| 70 | Arquitecturas y pruebas | 5 días | jue 08/09/16 | mié 14/09/16 |
| 71 | Generación de redes básicas | 3 días | jue 08/09/16 | lun 12/09/16 |
| 72 | Pruebas | 2 días | mar 13/09/16 | mié 14/09/16 |
| 73 | Arquitecturas contrastadas | 0 días | mié 14/09/16 | mié 14/09/16 |
| 74 | Gestión del proyecto | 184 días | lun 04/01/16 | jue 15/09/16 |
| 75 | **Reuniones de análisis** | **156 días** | **vie 22/01/16** | **vie 26/08/16** |
| 84 | Documentación | 184 días | lun 04/01/16 | jue 15/09/16 |
| 85 | | | | |

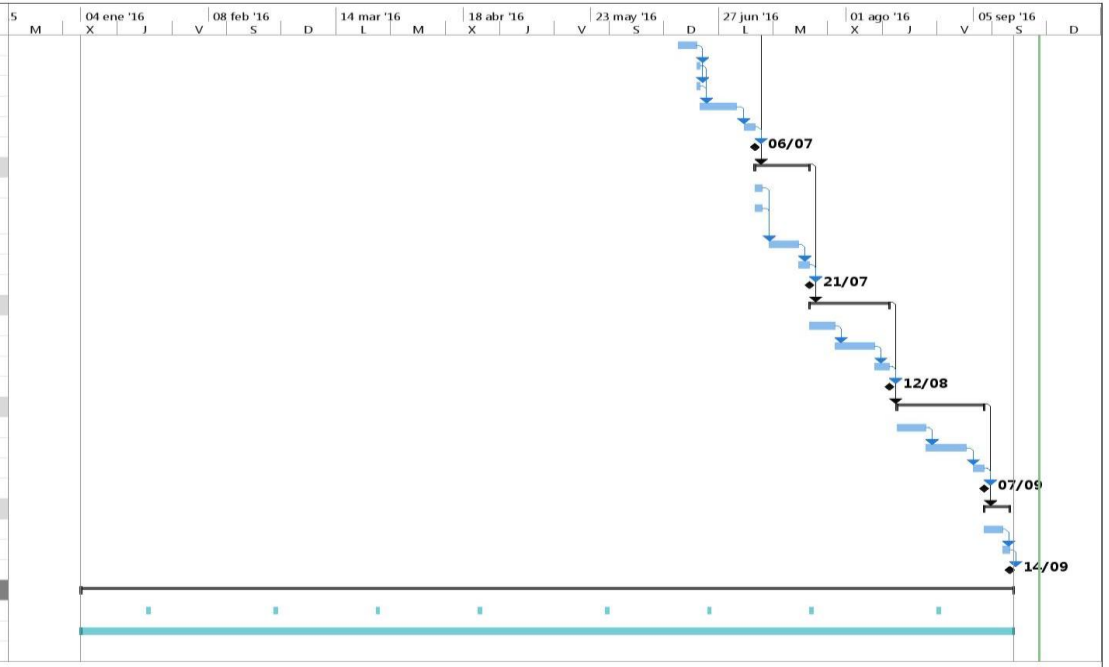


**Ilustración 8 - Diagrama gantt parte 2**

# 12 PRESUPUESTOS

En este apartado se expondrá un resumen de las partidas del presupuesto del proyecto

## 12.1 Horas internas

El equipo de trabajo ha sido descrito en el apartado 11.2, está compuesto por el director del proyecto, un diseñador, un desarrollador y un administrador. El coste por hora debido a cada cardo de los expuestos es el siguiente:

| CARGO | Coste/H (€) | Horas | Total (€) |
|---|---|---|---|
| Director de proyecto | 60 | 48 | 2880,00 |
| Diseñador | 30 | 153 | 4590,00 |
| Desarrollador | 30 | 377 | 11310,00 |
| Administrador | 20 | 202 | 4040,00 |
| TOTAL | | | 22820,00 |

**Tabla 6 - costes por hora de recursos**

## 12.2 Amortizaciones

Las necesidades materiales del proyecto se reducen a un equipo con capacidad de gestionar los entornos de desarrollo Mathematica y Omnet++ y el software de documentación

| MATERIAL | Coste | Amortización (meses) | Uso (meses) | Total (€) |
|---|---|---|---|---|
| Dell Optiplex 9020 MT + Pantalla | 700 | 36 | 9 | 175,00 |
| Windows 7 | 100 | 36 | 9 | 25,00 |
| Licencia Estudiante Mathematica | 0 | 0 | 9 | 0,00 |
| Licencia Omnet++ | 0 | 0 | 9 | 0,00 |
| Microsoft Office 2013 Professional | 150 | 36 | 9 | 37,50 |
| TOTAL | | | | 237,50 |

**Tabla 7 - costes por materiales**

## 12.3 Gatos

Debido a la naturaleza del proyecto a esta partida solo se asociarán gastos derivados de material de oficina. Según la estimación, se imputarán a este proyecto 100€ de material de oficina.

## 12.4 Costes indirectos

Tras el análisis de los gastos del laboratorio, se asigna un 10% del gasto a los costes indirectos, gastos realizados no imputables a ningún proyecto.

## 12.5 Resumen de costes

En este subapartado se expone el resumen del coste del proyecto llevado a cabo según el plan de trabajo expuesto en el apartado 11.

Tal como se ha analizado durante el apartado, el presupuesto se divide en diferentes partidas. A continuación se muestra una tabla con dicho desglose. Al ser un proyecto interno, no se añade el IVA.

| CONCEPTO | IMPORTE (€) |
|---|---|
| Horas internas | 22820,00 |
| Amortizaciones | 237,50 |
| Gastos | 100,00 |
| **SUBTOTAL** | **23157,50** |
| Costes indirectos (10%) | 2315,75 |
| **TOTAL** | **25473,25** |

**Tabla 8 - resumen de costes**

El importe total presupuestado para la realización del presente proyecto es de **25.473,25€**

# 13 CONCLUSIONES

El proyecto surge ante la necesidad de acelerar la familiarización de los estudiantes con las herramientas de desarrollo utilizadas en la asignatura Rendimiento en Redes de Telecomunicación. Esta ralentización disminuye la cantidad de comprobación y conceptos que el alumno puede tratar durante las prácticas.

La realización de este proyecto genera una serie de librerías que por un lado permiten al alumno realizar pruebas básicas de los conceptos tratados, por otro lado, puede servir como base y ejemplo para que el alumno realice sus propios desarrollos. La unión de estos beneficios implica la consecución del objetivo del proyecto reduciendo el tiempo de aprendizaje necesario para hacer uso de las herramientas de simulación.

Con el objetivo de poder ampliar las capacidades de las librerías en un futuro, ya sea por la ampliación de conceptos de la asignatura o por petición de un tercero, se han diseñado en función a estructuras simples.

Desde el punto de vista de producto, este proyecto crea herramientas didácticas y abiertas que permiten al usuario introducirse al análisis de redes en los entornos de desarrollo utilizados.

# 14 BIBLIOGRAFÍA

[1] A.Ferro Vazquez, L. Zabala Alberdi, Material de la asignatura "Rendimiento en Redes de Telecomunicación". Master en Ingeniería de Telecomunicación

[2] Omnet++ Simulation Manual, https://omnetpp.org/doc/omnetpp/manual/

[3] Mathematica documentation, http://reference.wolfram.com/language/

[4] Erlang, Agner Krarup *(1909)*. "The theory of probabilities and telephone conversations"

[5] Kendall, D. G. *(1953)*. "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain"

[6] RFC 1180 A TCP/IP Tutorial: https://tools.ietf.org/html/rfc1180

[7] RFC 3366 Advice to link designers on link Automatic Repeat reQuest (ARQ): https://tools.ietf.org/html/rfc3366

[8] International standard ISO/IEC 7498-1 Second Edition (1994)

# ANEXO I

## *PLIEGO DE CONDICIONES*

**Autor** Sayans Alberdi, Ainhoa

**Director** Ferro Vázquez, Armando

**Fecha** Septiembre 2016

# 1 ÍNDICE

# 2 INTRODUCCIÓN

En el presente documento se recogen las condiciones que el contratista y el desarrollador deberán aceptar para la correcta realización del proyecto. Entre las condiciones se incluyen tanto especificaciones técnicas, exigencias de calidad y normas que el producto debe, como las condiciones de garantía, entrega y mantenimiento.

Para asegurar el cumplimiento de las especificaciones, se desarrolla un plan de pruebas a realizar, el cual constituye las condiciones de aceptación del contratista. El producto debe pasar todos los exámenes realizados para ser aceptado. En el documento se deberán recoger todas las pruebas a realizar así como el escenario en el que se desarrollarán para poder describirlas en detalle.

# 3 CONDICIONES TÉCNICAS

## 3.1 Recursos materiales

En los siguientes apartados se enunciarán los recursos materiales necesarios en la realización del presente proyecto, se analizarán en dos categorías, recursos hardware y recursos software

### 3.1.1 Recursos hardware

Los recursos hardware a emplear para el correcto desarrollo del presente proyecto se constituyen por un único equipo de sobremesa cuyas características se describen a continuación:

- **Procesador:** 4th Generation Intel® Core™ i3 processor
- **RAM:** 4GB
- **Disco duro**: 500GB

Se imputa al proyecto exclusivamente los recursos hardware que tienen un tiempo de dedicación alto por parte del mismo. De este modo se excluyen equipos personales que miembros del grupo de trabajo puedan emplear en momento puntuales para labores del proyecto.

Todos los recursos anteriormente citados son propiedad de la Escuela Superior de Ingeniería de Bilbao con anterioridad al inicio del presente proyecto.

### 3.1.2 Recursos Software

A continuación se exponen los recursos software empleados para el correcto desarrollo del presente proyecto.

- **Sistema operativo:** Windows 7 Professional
- **Entorno ofimático:** Microsoft Office 2013 Professional
- **Entornos de desarrollo**
  - Mathematica 10.4
  - Omnet++
- **Gestor de versiones:** git

## 3.2 Documentación

En este apartado se enumeran los documentos que se deben entregar al futuro dueño del proyecto. Junto con la enumeración, se incluye una breve descripción del contenido de los mismos.

### 3.2.1 Documento 1: Memoria

En este documento se exponen las distintas razones que justifican la necesidad de realización del proyecto así como los pasos que se llevarán a cabo para su realización y los elementos necesarios para la misma.

Se presentará el contexto en el que se desarrollará el proyecto y se expondrán los objetivos del mismo. Además, se deberán analizar los diferentes beneficios que el proyecto aportará.

Se realizará el análisis de alternativas de ingeniería que se han valorado para dar soluciones de diseño válidas que permitan alcanzar los objetivos deseados. Tras esto, se realizará una descripción del diseño de alto nivel acorde a las alternativas seleccionadas.

A continuación se expondrán los datos necesarios para el seguimiento del proyecto. Para realizar esto, se analizarán la planificación y el presupuesto así como los métodos que se utilizan para el control del desarrollo del proyecto.

### 3.2.2 Anexo I: Pliego de condiciones

Este documento recoge las condiciones de entrega del proyecto, estas condiciones incluyen tanto las condiciones contractuales que tanto el contratista como el desarrollador deben aceptar como el plan de pruebas a llevar a cabo.

### 3.2.3 Anexo II: Diseño de bajo nivel

Este documento recoge el comportamiento de las funciones y módulos diseñados para el cumplimiento de los objetivos. Muestra en detalle el contenido de las diferentes librerías y las estructuras de datos utilizadas en las mismas.

### 3.2.4 Manual de uso

El documento deberá contener la explicación de uso de cada una de las librerías generadas. Se deberá detallar las entradas y salida de las distintas funciones y módulos. Se incluye la descripción y ejemplos de los ficheros de configuración

# 4 TAREAS A REALIZAR

En este apartado del documento se describen los diferentes paquetes de trabajo a realizar en el presente proyecto

## 4.1 PT.1-Definición del trabajo

En este paquete de trabajo se tiene como objetivo la definición de los objetivos que el proyecto debe cumplir. Durante este proceso se realizarán diversas reuniones con el grupo de trabajo.

Debido a que el proyecto completo se divide en necesidades individuales asociadas a los distintos conceptos tratados en la asignatura, este paquete se divide en tareas diferentes para cada uno de los mismos. Se obtendrán un documento recogiendo los diversos objetivos que se deben completar y el plan de pruebas de aceptación como objetivo final del paquete de trabajo.

## 4.2 PT.2-Librería de Mathematica para teoría de colas

Durante este paquete se tratarán todos los pasos necesarios para la realización de esta librería. Se estructura en subpaquetes:

### 4.2.1 PT.2.1-Diseño de alto nivel

En este subpaquete se abordan las decisiones del diseño de alto nivel de la librería. El primer paso será la realización del estado del arte de esta problemática en el entorno de desarrollo de su aplicación. Se establecerán las bases de uso de las funciones marcando la estructura general.

Como objetivo final se obtendrá el documento de diseño de alto nivel y el plan de pruebas de integración correspondiente.

### 4.2.2 PT.2.2-Diseño de bajo nivel

Este subpaquete trata de la distribución en funciones de los objetivos de la librería así como el funcionamiento de cada función. El resultado final de este apartado será el documento de diseño de bajo nivel y el plan de pruebas unitarias correspondiente a las diferentes funciones.

### 4.2.3 PT.2.3-Codificación y pruebas

En el transcurso de este subpaquete se deberá codificar y desarrollar el diseño llevado a cabo en los apartados anteriores del paquete de la librería. Al final de esta

sección se obtiene la librería completa que se entregará al contratista y su correspondiente manual.

## 4.3 PT.3-Librería de Mathematica para protocolos ARQ

Durante este paquete se tratarán todos los pasos necesarios para la realización de esta librería. Se estructura en los mismos subpaquetes que el paquete anterior por lo que no se describirán.

## 4.4 PT.4-Librería de Omnet++

En este paquete se tratarán los pasos necesarios para la realización de la librería de Omnet++. Al utilizarse la metodología en espiral para este desarrollo el mismo se dividirá en diferentes fases incrementales que partiendo de un diseño base permitirá la generación de módulos de manera sucesiva.

El resultado final de cada fase incluye los módulos y estructuras generadas así como el documento de diseño de los mismos. Una vez realizadas todas las fases se deberá generar el manual de uso de los módulos generados.

## 4.5 PT.5-Gestión del proyecto

En este paquete de trabajo se refleja la carga de trabajo derivada de la gestión general del proyecto Se trata de un paquete cuya extensión temporal es equivalente a la duración global del proyecto y engloba tareas como la gestión de los recursos destinados al proyecto, el control del avance de las tareas, el establecimiento de acciones de corrección en caso de ser necesarias, la revisión de entregables, etc. El control de todas estas tareas se realiza en las reuniones de análisis programadas periódicamente.

Están dentro de este paquete de trabajo la elaboración de las principales unidades entregables que son generadas para la realización del proyecto.

# 5 CONDICIONES DE RECPECIÓN

En este apartado se recogen las condiciones que deben cumplirse en la entrega del proyecto para que este sea validado. Se han definido una serie de pruebas que deberán ser realizadas para verificar los desarrollos y comprobar el correcto cumplimiento de las especificaciones acordadas.

Las pruebas recogidas en el plan constituyen la base para la aceptación del sistema. Por otro lado, no se elimina la posibilidad de pruebas complementarias realizadas por el equipo de desarrollo con el objetivo de realizar más comprobaciones.

## 5.1 Plazos de ejecución y entrega

En la siguiente tabla se muestra la fecha de los diferentes hitos del proyecto. Debido a posibles desvíos se admite una variación de ±4 días.

| Cód. | Librería | Entregable | Fecha |
|---|---|---|---|
| H.1 | | Alcance | 12/01/2016 |
| H.2 | | Plan de pruebas de aceptación | 12/01/2016 |
| H.3 | Librería de Mathematica de teoría de colas | Diseño de alto nivel | 18/01/2016 |
| H.4 | Librería de Mathematica de teoría de colas | Plan de pruebas de integración | 19/01/2016 |
| H.5 | Librería de Mathematica de teoría de colas | Diseño de bajo nivel | 02/02/2016 |
| H.6 | Librería de Mathematica de teoría de colas | Plan de pruebas unitarias | 03/02/2016 |
| H.7 | Librería de Mathematica de teoría de colas | Librería | 30/03/2016 |
| H.8 | Librería de Mathematica de protocolos ARQ | Diseño de alto nivel | 08/04/2016 |
| H.9 | Librería de Mathematica de protocolos ARQ | Plan de pruebas de integración | 11/04/2016 |
| H.10 | Librería de Mathematica de protocolos ARQ | Diseño de bajo nivel | 21/04/2016 |
| H.11 | Librería de Mathematica de protocolos ARQ | Plan de pruebas unitarias | 22/04/2016 |
| H.12 | Librería de Mathematica de protocolos ARQ | Librería | 15/06/2016 |
| H.13 | Librería de Omnet++ | Módulos Fase 1 | 06/07/2016 |
| H.14 | Librería de Omnet++ | Diseño Fase 1 | 06/07/2016 |
| H.15 | Librería de Omnet++ | Módulos Fase 2 | 21/07/2016 |
| H.16 | Librería de Omnet++ | Diseño Fase 2 | 21/07/2016 |
| H.17 | Librería de Omnet++ | Módulos Fase 3 | 12/08/2016 |
| H.18 | Librería de Omnet++ | Diseño Fase 3 | 12/08/2016 |
| H.19 | Librería de Omnet++ | Módulos Fase 4 | 07/09/2016 |
| H.20 | Librería de Omnet++ | Diseño Fase 4 | 07/09/2016 |
| H.21 | Librería de Omnet++ | Redes de prueba básicas | 14/09/2016 |
| H.22 | Librería de Omnet++ | Librería | 14/09/2016 |
| H.23 | | Manual | 15/09/2016 |

**Tabla 1 - hitos del proyecto**

Cualquier modificación en el plan de trabajo tras su aprobación por las dos partes, deberá ser discutida y aprobada por ambas, realizando las correspondientes

modificaciones en los documentos pertinentes y calculando el nuevo valor del presupuesto si este se viera modificado

## 5.2 Derechos de explotación sobre el producto

El proyecto será distribuido con licencia pública general de GNU. El equipo de trabajo y los contratistas conservarán los derechos de autor. Esta licencia posibilita la modificación y redistribución del software siempre bajo la misma licencia.

## 5.3 Pruebas de validación

En el apartado 9 de este documento, plan de validación, se define un plan de pruebas que deberá ser superado por el sistema desarrollado para considerarlo validado, en este punto el proyecto será entregado.

# 6 CONDICIONES ECONÓMICAS

En este apartado se exponen las condiciones económicas en las que se ha de realizar el proyecto, dividiendo estas en la especificación del coste total del proyecto y la forma de pago del mismo.

## 6.1 Coste del proyecto

El importe total presupuestado para la realización del presente proyecto asciende a un total de **veinticinco mil cuatrocientos setenta y tres euros y veinticinco céntimos (25.473,25€)** impuestos no incluidos.

## 6.2 Forma de pago

Se plantea el pago en 3 etapas. En el momento de aceptación del contrato se realizará un pago por valor de 5.000,25€ por parte de la empresa contratante en concepto de conformidad del mismo. Posteriormente, se realizará un segundo pago 10.236,50€ en un plazo de cinco meses desde la recepción del primer pago. Finalmente se realizará un tercer pago con un importe de 10.236,50€ a la entrega del trabajo realizado.

# 7 CONDICIONES CONTRACTUALES

## 7.1 Actas de recepción

La recepción provisional del producto se efectuará una vez validado el mismo, debiendo firmar tanto el proyectista como el contratista el Acta de Recepción Provisional. Partiendo de la creación de dicho acta comenzará el plazo de reclamación, cuya duración será de dos semanas. Transcurrido este plazo, y a no ser que el contratista notifique los defectos para su subsanación dentro del mismo, se suscribirá el Acta de Recepción Definitiva.

## 7.2 Responsabilidades del cliente

El cliente se responsabiliza del cumplimiento de toda normativa legal vigente relativa a los servicios contratados, incluyendo los premisos y licencias de utilización, así como la normativa de propiedad intelectual y de confidencialidad del desarrollo realizado.

## 7.3 Responsabilidad del proyectista

El presente proyecto se basa en la realización de un diseño que resuelva las necesidades del cliente así como del desarrollo de este diseño en un producto final.

El diseñador se responsabiliza de los posibles fallos de diseño que se puedan dar en ese producto. Sin embargo, no se responsabiliza de los errores en el desarrollo del mismo ni en el rediseño de alguno de sus módulos.

Por el contrario, el desarrollador se responsabiliza de la realización del diseño y de los posibles cambios en el mismo durante el desarrollo

## 7.4 Extinción del contrato

El contrato se extinguirá bien por conclusión o complimiento del mismo, o por resolución de una de las partes involucradas en el mismo. Serán consideradas causas de resolución:

- El incumplimiento de las cláusulas contenidas en el Pliego de Condiciones.
- La extinción de la personalidad jurídica de la sociedad mercantil de una de las partes, salvo que el patrimonio sea incorporado a otra entidad.
- Mutuo acuerdo entre las partes
- La declaración de quiebra o suspensión de pagos de una de las partes

## 7.5 Resolución de conflictos

Los litigios que puedan surgir sobre la interpretación o modificación del contrato serán resueltos por los Juzgados y Tribunales, renunciando a cualquier otro fuero que pudiera corresponder a las partes. Por tanto, se encomienda a estos el nombramiento de árbitro y la administración del arbitraje cuyo fallo las partes se comprometen aceptar.

# 8 ASPECTOS JURÍDICOS

Se contemplan los siguientes aspectos jurídicos relacionados con la contratación y el desarrollo del proyecto.

## 8.1 Fuerza mayor

El contratista no será considerado responsable por el incumplimiento de las obligaciones en tanto la ejecución de los trabajo se retrasase o se hiciese imposible a causa de fuerza mayor.

Son consideradas causas de fuerza mayor todos aquellos sucesos o circunstancias, fuera de control del contratista o del comprador y cualesquiera otras circunstancias que fueran imprevisibles, o que siendo previsibles fueran inevitables, de acuerdo con la jurisprudencia y la doctrina legal sentada sobre este concepto en el Código Civil.

Las antedichas causas de fuerza mayor se tomarán en consideración únicamente cuando afecten directamente al suministro.

## 8.2 Arbitrajes y tribunales

Tanto el comprador como el contratista se comprometen a cumplir las condiciones establecidas en el presente Pliego de Condiciones y en la documentación complementaria del contrato, resolviendo por medio de acuerdos y negociaciones las posibles diferencias que puedan surgir entre ellos respecto a la aplicación, desarrollo, cumplimiento, ejecución o interpretación de los mismos.

En caso de que cualquier posible discrepancia o controversia entre ellos no pudiese ser llevaba a buen fin, resolviéndose con éxito en la forma anteriormente indicada, el contratista se compromete a someter tales diferencias a arbitraje, formalizado de acuerdo a las normas reguladoras del mismo, contenidas en la vigente Ley de Arbitraje de Derecho Privado.

El arbitraje se realizaré en Bilbao y en la escritura ha de figurar el compromiso de plazo de 30 días como término en que los árbitros han de pronunciar el fallo correspondiente.

# 9 PLAN DE VALIDACIÓN

En este apartado se recogen las pruebas más importantes a realizar para verificar el correcto funcionamiento del sistema. Estas pruebas son las siguientes:

- Prueba I: teoría de colas en Mathematica
- Prueba II: trasmisión con protocolos ARQ en Mathematica
- Prueba III: trasmisión con protocolos ARQ en Omnet++
- Prueba IV: simulación de redes Jacksonianas en Omnet++
- Prueba V: simulación de redes de rutado fijo en Omnet++
- Prueba VI: simulación de multiplexación estadística en Omnet++

Los siguientes apartados recogen, por un lado, el escenario en el que se desarrollar las pruebas anteriormente mencionadas para continuar describiendo cada una de ellas en detalle.

## 9.1 Arquitectura del escenario

Para hacer uso de las herramientas desarrolladas solo es necesario un equipo capaz de hacer uso de los entornos de desarrollo para los que se generan las librerías. Antes esta situación las pruebas se realizarán en el mismo equipo de desarrollo. En ambos sistemas se instalarán las librerías desarrolladas.

## 9.2 Especificación de las pruebas

### 9.2.1 Prueba I: teoría de colas en Mathematica

#### 9.2.1.1 Descripción de la prueba

Con esta prueba se pretende verificar la capacidad de la librería generada para modelar todo tipo de colas en el entorno de desarrollo Mathematica.

#### 9.2.1.2 Procedimiento

Para la realización de esta prueba se creará un proyecto de Mathematica que cargue la librería que se desea probar. Se realizarán comprobaciones de cada una de las funciones de la librería siguiendo el funcionamiento diseñado

- Generación de paquetes
- Modulación de colas G/G/1
- Modulación de colas G/G/2
- Modulación de colas G/G/1/N

- Simulación completa
- Cálculo de probabilidad de estado
- Cálculo de tiempo medio de estado en el sistema
- Cálculo del Throughput.

Para validar esta prueba se realizarán dos tipos de comprobaciones. Por un lado, se analizarán que los paquetes generados cumplen las condiciones establecidas por parámetros y se comprobará que el comportamiento de cada función es el esperado con entradas controladas. Por otro lado se comprobará que los resultados obtenidos coinciden con los resultados teóricos.

### 9.2.2 Prueba II: trasmisión con protocolos ARQ en Mathematica

#### 9.2.2.1 Descripción de la prueba

Estas comprobaciones tienen como objetivo verificar la correcta simulación de transmisión de datos mediante protocolos ARQ en el entorno de desarrollo de Mathematica mediante la librería generada con dicho propósito.

#### 9.2.2.2 Procedimiento

Al igual que las pruebas anteriores en este caso se creará un proyecto de Mathematica en el que se cargará la librería que se desea comprobar y la librería "drawTX.m" con la que debe ser compatible.

A continuación se enumeran las comprobaciones que se deben realizar.

- Generación de paquetes
- Simulación de transmisión según Stop & Wait
- Simulación de transmisión según Go Back N
- Simulación completa
- Compatibilidad con librería "drawTX.m"

La prueba se dará por validada cuando las simulaciones se adecuen a los protocolos tratados y esta se pueda mostrar gráficamente mediante la librería "drawTX.m".

### 9.2.3 Prueba III: trasmisión con protocolos ARQ en Omnet++

#### 9.2.3.1 Descripción de la prueba

En este caso se deberán realizar las simulaciones de los protocolos ARQ pero en Omnet++ y comprobar que los resultados son acordes a la librería de Mathematica. Se

deberá comprobar que los protocolos son aplicables tanto a nivel de enlace como extremo a extremo.

#### 9.2.3.2 Procedimiento

Para realizar las comprobaciones se generarán diferentes redes que hagan uso de los módulos generados comprobando cada uno de ellos. Para realizar las evaluaciones a nivel de enlace se realizarán comunicación con dos host mientras que para evaluar la gestión extremo a extremo se utilizarán varios equipos intermedios.

- Trasmisión Stop & Wait a nivel de enlace
- Trasmisión Go Back N a nivel de enlace
- Trasmisión Stop & Wait extremo a extremo
- Trasmisión Go Back N extremo a extremo

Se considerarán lo módulos validados cuando el comportamiento de los mismo sea el esperado en función del protocolo. En el caso del nivel de enlace se considera necesario comprobar la concordancia con la librería del mismo propósito de Mathematica.

Junto con esto, se deberá asegurar que cada módulo extrae los datos que se espera obtener así como que los datos son correctos.

### 9.2.4 Prueba IV: simulación de redes Jacksonianas en Omnet++

#### 9.2.4.1 Descripción de la prueba

Con esta prueba se quiere evaluar el correcto funcionamiento del módulo de rutado en base a probabilidades.

#### 9.2.4.2 Procedimiento

Para realizar estas comprobaciones se generarán diversas redes sin error en los enlaces en las que los módulos de rutado hagan uso de la conmutación según probabilidad. En cada una de las pruebas se seguirán los siguientes pasos:

- Creación de la red en lenguaje NED
- Análisis matemático de la red creada
- Comprobación de los resultados

Este procedimiento se deberá repetir en seis ocasiones. En cada una de las pruebas se inyectarán al menos 100.000 paquetes para disminuir el efecto de la aleatoriedad.

El funcionamiento se dará por válido cuando los resultados obtenidos sigan los teóricos con una posible desviación de ±0.5%. También se debe asegurar que los diferentes módulos emiten los datos estadísticos asociados a los mismos.

### 9.2.5 Prueba V: simulación de redes de rutado fijo en Omnet++

#### 9.2.5.1 Descripción de la prueba

El objetivo de esta prueba es la comprobación del correcto funcionamiento del módulo de rutado en base a una tabla de rutado precargada.

#### 9.2.5.2 Procedimiento

El procedimiento seguido en esta prueba es el mismo que el caso anterior:

- Creación de la red en lenguaje NED
- Análisis matemático del tráfico en los enlaces
- Comprobación de los resultados

Este procedimiento se deberá repetir en seis ocasiones. En cada una de las pruebas se deberán establecer al menos 3 generadores de tráfico y 3 receptores. Debido a que en este el comportamiento no dispone de aleatoriedad el resultado debe ser exactamente el esperado.

El funcionamiento se dará por válido cuando los resultados concuerden con los cálculos. Del mismo modo se deberá comprobar la extracción de información de cada uno de los módulos utilizados.

### 9.2.6 Prueba VI: simulación de multiplexación estadística en Omnet++

#### 9.2.6.1 Descripción de la prueba

En estas pruebas se comprobará que los módulos asociadlos a la multiplexación de paquete de varias líneas en una sola línea.

#### 9.2.6.2 Procedimiento

Para llevar a cabo estas comprobaciones se realizarán dos redes, una que compruebe la multiplexación estadística básica y otra con la multiplexación con prioridad. Se crearán dos redes, una basada en la multiplexación estadística y otra con multiplexación con prioridad.

Se darán por validados los módulos cuando el comportamiento sea el configurado y la extracción de datos sea correcta.

## 9.3 Actas de realización de las pruebas

| ACTA DE REALIZACIÓN DE PRUEBA I | | | |
|---|---|---|---|
| Información básica | | | |
| Operario: | | Firma: | |
| Fecha: | | | |
| Hora: | | | |
| Formulario de funcionamiento | | | Estado |
| Generación de paquetes | | | |
| Modelación de cola G/G/1 | | | |
| Modelación de cola G/G/2 | | | |
| Modelación de cola G/G/1/N | | | |
| Modelado completo | | | |
| Cálculo de probabilidad de estado | | | |
| Cálculo de tiempo medio de estado en el sistema | | | |
| Cálculo de Throughput | | | |
| Incidencias de generación de paquetes | | | |
| | | | |
| Incidencias de modelo | | | |
| | | | |
| Incidencias de extracción de datos | | | |
| | | | |
| Comentarios | | | |
| | | | |

**Tabla 2 - acta de realización de prueba I**

| ACTA DE REALIZACIÓN DE PRUEBA II | | | |
|---|---|---|---|
| Información básica | | | |
| Operario: | | Firma: | |
| Fecha: | | | |
| Hora: | | | |
| Formulario de funcionamiento | | | Estado |
| Generación de paquetes | | | |
| Simulación de protocolo Stop & Wait | | | |
| Simulación de protocolo Go Back N | | | |
| Simulación completa | | | |
| Muestra con librería “drawTX.m” | | | |
| Incidencias de generación de paquetes | | | |
| | | | |
| Incidencias de simulación | | | |
| | | | |
| Comentarios | | | |
| | | | |

**Tabla 3 - acta de realización de prueba II**

| ACTA DE REALIZACIÓN DE PRUEBA III | | | |
|---|---|---|---|
| Información básica | | | |
| Operario: | | Firma: | |
| Fecha: | | | |
| Hora: | | | |
| Formulario de funcionamiento | | | Estado |
| Stop & Wait a nivel de enlace | | | |
| Go Back N a nivel de enlace | | | |
| Stop & Wait extremo a extremo | | | |
| Go Back N extremo a extremo | | | |
| Incidencia a nivel de enlace | | | |
| | | | |
| Incidencias extremo a extremo | | | |
| | | | |
| Comentarios | | | |
| | | | |

**Tabla 4 - acta de realización de prueba III**

| ACTA DE REALIZACIÓN DE PRUEBA IV | | | |
|---|---|---|---|
| Información básica | | | |
| Operario: | | Firma: | |
| Fecha: | | | |
| Hora: | | | |
| Prueba 1 | | | |
| Hosts: | | Resultado: | |
| Prueba 2 | | | |
| Hosts: | | Resultado: | |
| Prueba 3 | | | |
| Hosts: | | Resultado: | |
| Prueba 4 | | | |
| Hosts: | | Resultado: | |
| Prueba 5 | | | |
| Hosts: | | Resultado: | |
| Prueba 6 | | | |
| Hosts: | | Resultado: | |
| Incidencias | | | |
| | | | |
| Comentarios | | | |
| | | | |

**Tabla 5 - acta de realización de prueba IV**

| ACTA DE REALIZACIÓN DE PRUEBA V | | | |
|---|---|---|---|
| Información básica | | | |
| Operario: | | Firma: | |
| Fecha: | | | |
| Hora: | | | |
| Prueba 1 | | | |
| Hosts: | | Resultado: | |
| Prueba 2 | | | |
| Hosts: | | Resultado: | |
| Prueba 3 | | | |
| Hosts: | | Resultado: | |
| Prueba 4 | | | |
| Hosts: | | Resultado: | |
| Prueba 5 | | | |
| Hosts: | | Resultado: | |
| Prueba 6 | | | |
| Hosts: | | Resultado: | |
| Incidencias | | | |
| | | | |
| Comentarios | | | |
| | | | |

**Tabla 6 - acta de realización de prueba V**

| ACTA DE REALIZACIÓN DE PRUEBA VI | | | |
|---|---|---|---|
| Información básica | | | |
| Operario: | | Firma: | |
| Fecha: | | | |
| Hora: | | | |
| Formulario de funcionamiento | | | Estado |
| Multiplexación estadística | | | |
| Multiplexación estadística con prioridad | | | |
| Incidencia de multiplexación | | | |
| | | | |
| Incidencias de demultiplexación | | | |
| | | | |
| Comentarios | | | |
| | | | |

**Tabla 7 - acta de realización de prueba VI4**

# ANEXO II

## *DISEÑO DE BAJO NIVEL*

**Autor** Sayans Alberdi, Ainhoa

**Director** Ferro Vázquez, Armando

**Fecha** Septiembre 2016

# 1 ÍNDICE

# 2 LISTA DE ILUSTRACIONES

# 3 INTRODUCCIÓN

En este anexo se analizará detalladamente el diseño de los diversos módulos e interfaces analizadas en el diseño de alto nivel. Al igual que en el punto anterior se analizará cada una de las tres librerías de manera independiente.

- Librería de Mathematica para colas G/G/X
- Librería de Mathematica para transmisiones ARQ
- Librería de Omnet++

# 4 LIBRERÍA DE MATHEMATICA PARA COLAS G/G/X

Las funciones y estructuras diseñadas para esta librería tienen como objetivo la modelación de colas del tipo G/G/X.

## 4.1 Estructuración de paquetes

El primer paso para diseñar la librería es decidir la representación de los paquetes en la misma. Mathematica no permite la definición de estructuras como tal pero permite trabajar con arrays y matrices, arrays de arrays. Ante esta situación se decide que un array de longitud determinada y una matriz con los mismos representarán una lista de paquetes.

Esta será la estructura seguida:

1. **Tiempo de llegada a la cola**: momento temporal en el que llega al sistema
2. **Tiempo de entrada en el servidor**: momento temporal de entrada al procesador
3. **Tiempo de procesado**: tiempo total de procesado
4. **Secuencia**: número de secuencia del paquetes

En caso de que el paquete sea descartado porque la cola está llena, casuística posible en el caso de las colas finitas, el segundo parámetro, tiempo de entrada en el servidor, será nulo.

## 4.2 Funciones

A continuación se analizarán las funciones, su objetivo, su entrada y salida. De acuerdo con los objetivos establecidos para esta librería se dividirán las funciones en dos grupos.

- **Simulación:** genera paquetes y permite realizar su procesado. Debe permitir hacerlo paso a paso y directamente sin necesidad de conocer el funcionamiento interno.
- **Resultados y visualización:** grupo de funciones para la extracción de resultados y la correcta visualización de los mismos.

### 4.2.1 Simulación

En este apartado se analizarán las diferentes funciones generadas para modular las distintas colas.

#### 4.2.1.1 ArrivalAcumSeries

El objetivo de esta función es la generación de la lista de paquetes de llegada a la cola.

- Entrada
    - **Lmb:** Tiempo entre llegadas. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
    - **Mu:** Tiempo de procesado. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
    - **N**: número de paquetes a generar.
- Salida
    - Lista de paquetes que se introducirán en una cola

#### 4.2.1.2 FifoDepartureTime1

Modela una cola G/G/1. El tipo de cola será definida por la distribución de tiempo entre paquetes y la distribución de tiempo de procesado de los mismos de la lista de paquetes de entrada.

- Entrada
    - **Arrivals:** lista de paquetes de llegada
- Salida
    - Lista de paquetes modificados tras ser procesados por la cola

#### 4.2.1.3 Queue1Sim

Simula completamente la generación y procesado de paquetes de una cola G/G/1. El objetivo de esta función es permitir al usuario realizar una simulación completa. En este caso el tipo de cola será establecido por los propios parámetros de la función.

- Entrada
    - **Lmb:** Tiempo entre llegadas. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
    - **Mu**: Tiempo de procesado. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
    - **N:** Cantidad de paquetes que se procesarán

- Salida
    - Lista de paquetes procesados

#### 4.2.1.4 FifoDepartureTime2

Aplica una cola G/G/2. El tipo de cola será establecida por la distribución de tiempo entre paquetes y distribución de tamaño de paquetes de la lista de entrada.

- Entrada
    - **Arrivals:** lista de paquetes que llegan a la cola
- Salida
    - Lista de paquetes procesados por los dos servidores.

#### 4.2.1.5 Queue2Sim

Realiza la simulación completa de generación y procesado de paquetes por una cola G/G/2. Esta función permite simular por completo este tipo de cola sin conocer los detalles de la generación de paquetes. Los distintos tipos de cola vendrán declarados según los parámetros de entrada.

- Entrada
    - **Lmb:** Tiempo entre llegadas. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
    - **Mu**: Tiempo de procesado. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
    - **N:** Cantidad de paquetes que se procesarán
- Salida
    - Lista de paquetes procesados

#### 4.2.1.6 FifoDepartureTime1N

Aplica una cola G/G/1/N. El tipo de cola vendrá identificado por la distribución de tiempo entre paquetes y distribución de tamaño de paquetes de la lista de paquetes de entrada.

- Entrada

- **Arrivals:** lista de paquetes que llegan a la cola
- **Length**: longitud del sistema, longitud de la cola +1

- Salida
  - Lista de paquetes procesados por el servidor e identificados los paquetes que han sido descartado por la cola

#### 4.2.1.7 Queue1SimN

Simula por completo una cola G/G/1/N. El objetivo de la función es la simulación completa de la cola según los parámetros que la identifican. Las características de la cola vendrán determinada según los parámetros de entrad.

- Entrada
  - **Lmb:** Tiempo entre llegadas. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
  - **Mu**: Tiempo de procesado. Permitirá la introducción de una función que se evaluará cada vez que sea necesario. Esto permitirá la introducción de cualquier distribución aleatoria.
  - **N:** Cantidad de paquetes que se procesarán
  - **Length:** longitud del sistema, longitud de la cola +1

- Salida
  - Lista de paquetes procesados y descartados

### 4.2.2 Resultados y visualización

Las funciones que se mostrarán a continuación permiten al usuario la extracción de datos de las modulaciones sin necesidad de conocer la estructura de los paquetes.

#### 4.2.2.1 Stire

Esta función devolverá la versión escalonada de una array del tipo {X,Y}. Muchas de las estadísticas extraídas serán una evolución temporal de variables discretas. La evolución de estas variables se muestra mejor en diagramas escalonados que muestren el valor real del estadístico a lo largo del tiempo y no el punto de cambio.

- Entrada

- **Array:** array de elementos {X,Y}

- Salida

  - Versión escalonada del array de entrada.

#### 4.2.2.2 StatusAction

Esta función tiene como objetivo determinar el momento de cambio de estado del sistema y el sentido del cambio, positivo o negativo.

- Entrada

  - **Array:** lista de paquetes ya procesados por una cola

- Salida

  - Array de elementos {t,a} donde 't' es el momento temporal en el que ocurre un cambio de estado en el sistema y 'a' será el cambio ocurrido, +1/-1. El array está ordenado temporalmente.

#### 4.2.2.3 StateTime

La función evalúa el estado del sistema en cada momento.

- Entrada

  - **Array**: lista de paquetes ya procesada por una cola

- Salida

  - Array de elementos {t,s} done 't' es el momento temporal en el que ocurre un cambio de estado y 's' el estado que tendrá a partir de dicho instante. El array está ordenado temporalmente.

#### 4.2.2.4 StatusProbPasta

Calcula la probabilidad de encontrar el sistema en cada estado según la probabilidad pasta de las distribuciones poissonianas. Esta función solo tendrá sentido en colas M/G/X.

- Entrada

  - **Array**: lista de paquetes ya procesada por una cola

- Salida

- Array de elementos {s,p} donde 's' es el estado del sistema y 'p' la probabilidad de que el sistema esté en el mismo. El array está ordenado según los estados de manera ascendente.

#### 4.2.2.5 StatusProbTime

Evalúa la probabilidad de encontrar el sistema en cada posible estado según el tiempo que ha estado en el mismo.

- Entrada
  - **Array:** lista de paquetes procesado por una cola
- Salida
  - Array de elementos {s,p} donde 's' es el estado del sistema y 'p' la probabilidad de que el sistema esté en el mismo. El array está ordenado según los estados de manera ascendente.

#### 4.2.2.6 MeanSystemtime

La función calcula el tiempo medio de estado en el sistema para los paquetes procesado.

- Entrada
  - **Array:** lista de paquetes procesado por una cola
- Salida
  - Devuelve el tiempo medio de estado en el sistema según los paquetes procesados.

#### 4.2.2.7 QueueThAcum

El objetivo de la función es calcular el Throughput de la cola.

- Entrada
  - **Array:** lista de paquetes procesados por una cola
- Salida
  - Devuelve un array del tipo {t,th} donde 't' es el tiempo y 'th' el Throughput en dicho instante.

# 5 LIBRERÍA DE MATHEMATICA DE PROTOCOLOS ARQ

Cómo se ha indicado en las especificaciones la estructura de paquetes utilizada por esta librería debe ser compatible con la librería "drawTX.m". En este caso las funciones solo deberán realizar la simulación de los protocolos, queda a cargo del alumno la realización de las funciones para la extracción de estadísticos.

## 5.1 Estructuración de paquetes

La estructura de los paquetes utilizados en la librería "drawTX.m" es la siguiente:

1. **Tiempo de llegada:** momento en que el paquete llega al servidor o momento de comienzo de transmisión según si la lista ha sido ya enviada o no.
2. **Tiempo de inserción (o tamaño de paquete):** tiempo que se tarda en transmitir el paquete. En la librería generada se trabajará con tamaños de paquete y luego se adaptará.
3. **Número de secuencia:** secuencia del paquete dentro de la transmisión
4. **Error:** si el paquete es erróneo o no. Indica el tipo de representación que se realizará en la librería.
5. **Número de repetición:** repetición del paquete

## 5.2 Funciones

A continuación se indicarán las funciones creadas para poder simular los protocolos Stop & Wait y Go Back N.

### 5.2.1 SetIniPar

Tiene como objetivo inicializar los parámetros de la librería de simulación. No afecta a los parámetros de la librería de visualización.

- Entrada
  - **Speed:** velocidad de la línea en bits/s
  - **Propagation time:** tiempo de propagación de los paquetes de un nodo al otro, en segundos
  - **Ack Time:** tiempo de inserción de los paquetes ACK – NACK, en segundos.
- Salida
  - Array de tres elementos donde cada uno es uno de los parámetros de entrada en el mismo orden.

### 5.2.2 GetPacketArrival

Genera los paquetes que se transmitirán mediante los diversos protocolos.

- Entrada
    - **Lmb**: Tiempo entre llegadas. Se evaluarán cada vez que sean utilizadas para permitir la introducción de funciones generadoras de distribuciones aleatorias. Los valores se introducen en segundos.
    - **Mu**: tiempo de inserción del paquete. Se evaluará cada vez que sean utilizadas para permitir la introducción de funciones generadores de distribuciones aleatorias. Los valores se introducen en segundos.
    - **InitTime:** tiempo en el que comienzan a generarse los paquetes
    - **N:** número de paquetes a generar
- Salida
    - Array de paquetes generados según los parámetros. El segundo parámetro representará el tamaño de paquete. El cuarto y quinto paquete tendrán valor 0

### 5.2.3 FifoSW

Simula la trasmisión de una lista de paquetes según los parámetros de la librería mediante el protocolo Stop & Wait.

- Entrada
    - **Lpkt:** lista de paquetes que se deben transmitir
    - **Error:** probabilidad de error de la línea, un valor entre 0 y1
- Salida
    - Lista de paquetes modificada según la simulación de la transmisión. El segundo parámetro representará el tamaño de paquete.

### 5.2.4 ToTime

Convierte los paquetes en los que el segundo parámetro representa el tamaño de paquete a segundos según la velocidad establecida en la librería.

- Entrada

- **Pkt**: lista de paquetes en la cual el segundo parámetro indica el tamaño del paquete

- Salida

  - La lista de paquete de entrada a la que se le ha modificado el segundo parámetro y se representa en segundos.

### 5.2.5 LaunchSimTxSW

Realiza la simulación completa de la transmisión de paquetes con el protocolo Stop & Wait según los parámetros indicados. Los parámetros de la línea serán los de la simulación al final de la misma.

- Entrada

  - **Lmb:** Tiempo entre llegadas. Se evaluarán cada vez que sean utilizadas para permitir la introducción de funciones generadoras de distribuciones aleatorias. Los valores se introducen en segundos.

  - **Tp:** Tiempo de propagación de los paquetes entre los equipos.

  - **P:** probabilidad de error en la línea

  - **N:** número de paquete que se simularán

  - **Mu:** tiempo de inserción del paquete. Se evaluará cada vez que sean utilizadas para permitir la introducción de funciones generadores de distribuciones aleatorias. Los valores se introducen en segundos.

  - **Speed:** velocidad de la línea

  - **Ack:** tiempo de inserción de los ACK o NACK

- Salida

  - Devuelve una lista de paquetes que simulan la transmisión en función del protocolo Stop & Wait. El segundo parámetro de los paquetes representará el tiempo de inserción en segundos. La simulación de estos paquetes comenzará en el segundo 0.

### 5.2.6 FifoGBN

Simula la trasmisión de una lista de paquetes según los parámetros de la librería mediante el protocolo Go Back N.

- Entrada
    - **Lpkt:** lista de paquetes que se deben transmitir
    - **Error:** probabilidad de error de la línea, un valor entre 0 y1
- Salida
    - Lista de paquetes modificada según la simulación de la transmisión. El segundo parámetro representará el tamaño de paquete.

### 5.2.7 LaunchSimTxGBN

Realiza la simulación completa de la transmisión de paquetes con el Go Back N según los parámetros indicados. Los parámetros de la línea serán los de la simulación al final de la misma.

- Entrada
    - **Lmb:** Tiempo entre llegadas. Se evaluarán cada vez que sean utilizadas para permitir la introducción de funciones generadoras de distribuciones aleatorias. Los valores se introducen en segundos.
    - **Tp:** Tiempo de propagación de los paquetes entre los equipos.
    - **P:** probabilidad de error en la línea
    - **N:** número de paquete que se simularán
    - **Mu:** tiempo de inserción del paquete. Se evaluará cada vez que sean utilizadas para permitir la introducción de funciones generadores de distribuciones aleatorias. Los valores se introducen en segundos.
    - **Speed:** velocidad de la línea
    - **Ack:** tiempo de inserción de los ACK o NACK
- Salida
    - Devuelve una lista de paquetes que simulan la transmisión en función del protocolo Go Back N. El segundo parámetro de los paquetes representará el tiempo de inserción en segundos. La simulación de estos paquetes comenzará en el segundo 0.

# 6 LIBRERÍA DE OMNET++

En este apartado se analizarán los distintos módulos introducidos en el diseño de alto nivel. Al igual que en dicho apartado se realizará un análisis por capas. La distribución de capas se ha basado en la estructura básica descrita en el RFC 1180.

La siguiente imagen muestra las capas y como se deben poner interconectar entre ellas.

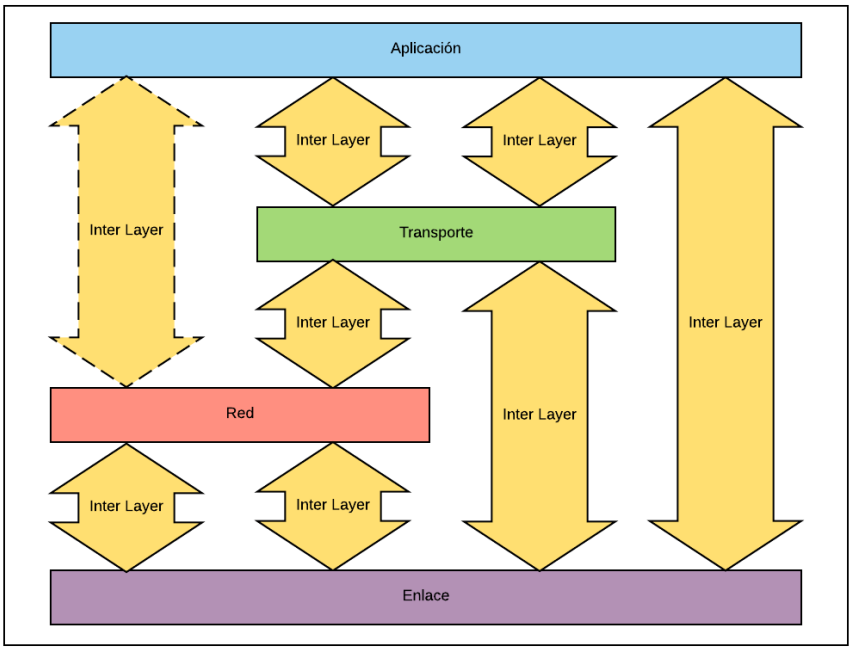


**Ilustración 1 - Distribución e interconexión de capas**

La conexión entre capas dentro de un mismo equipo se conectará mediante canales básicos de Omnet++ que se denominarán 'M' a partir de ahora. Por el contrario, las conexiones entre equipos se realizarán mediante conexiones del tipo 'ned.DatarateChannel' que a partir de este momento se denominarán 'C'.

Los canales M que se utilizan para conectar los distintos módulos de un mismo equipo comunicarán paquetes inter_layer según lo especificado en la interfaz.

Todos los mensajes intercambiados entre los distintos módulos y equipos heredarán de la clase cPacket de Omnet++.

## 6.1 Interfaz Inter_Layer

En este apartado se analizará la interfaz de comunicación entre las distintas capas. Esta interfaz trata de un paquete que contiene los datos necesarios de comunicación así como encapsular el paquete de la capa de menor nivel.

Se utilizará la abreviatura IL para referirse a este tipo de paquetes

### 6.1.1 Datos de la cabecera

El paquete de comunicación se denominará inter_layer. La cabecera tendrá los siguientes datos:

- Origen
- Destino
- Protocol

La interpretación de cada uno de los parámetros varía en función de si el uso del paquete es en dirección ascendente o descendente

### 6.1.2 Funcionamiento ascendente

En este caso es la capa inferior quien indica a la capa superior que se ha recibido un paquete para ella. El paquete encapsulado será el de la capa superior. Esta es la interpretación de los datos de cabecera:

- **Origen:** dirección origen del paquete recibido en la capa inferior
- **Destino:** dirección destino del paquete recibido en la capa inferior
- **Protocol** no es necesario en este sentido

### 6.1.3 Funcionamiento descendente

En este caso es la capa superior la que indica a la capa inferior que tiene datos que transmitir. El paquete encapsulado por la interfaz será el de la capa superior. A continuación se muestra la interpretación de los datos de la cabecera

- **Origen**: dirección origen del paquete de la capa superior
- **Destino**: dirección destino de la capa inferior
- **Protocol**: identificador de capa que genera el paquete

## 6.2 Direccionamiento

Para establecer el modelo de direccionamiento se han utilizado las referencias básicas descritas en estándar internacional ISO/IEC 7498-1 realizando ciertas modificaciones.

Al igual que se indica en el estándar solo se establecerán direcciones para la capa de transporte y red. La capa de enlace no requiere de direccionamiento ya que se utilizarán protocolos punto a punto.

Las direcciones de red estarán compuestas por números enteros entre 0 y 99. Las direcciones de la capa de transporte por su parte están compuestas por números de 3 dígitos, los dos primeros identifican la dirección de red del host y el tercero el puerto de interconexión entre la capa de red y el módulo de transporte de ese mismo equipo. La ilustración 2 muestra esta disposición.

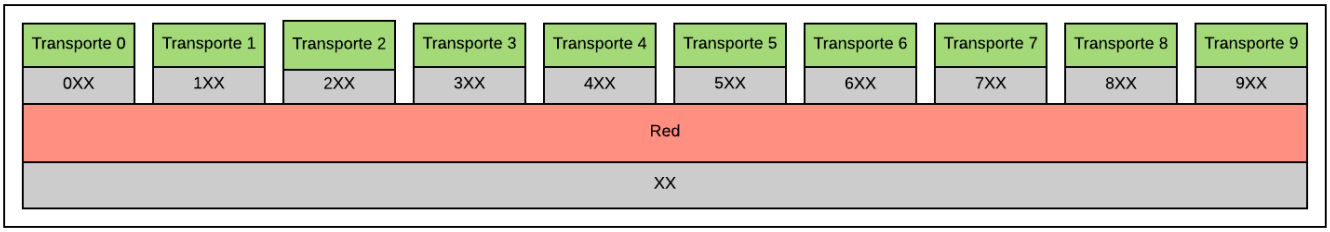


**Ilustración 2 - Distribución de direcciones en un equipo**

## 6.3 Aplicación

Esta capa tiene el objetivo de generar y recibir tráfico de datos. A continuación se analizará el paquete utilizado en esta capa así como los módulos utilizados. A continuación se enumeran los módulos que se utilizarán y un diagrama con su uso:

- **Injector:** módulo encargado de la generación del tráfico
- **Dump:** módulo encargado de la recogida del tráfico

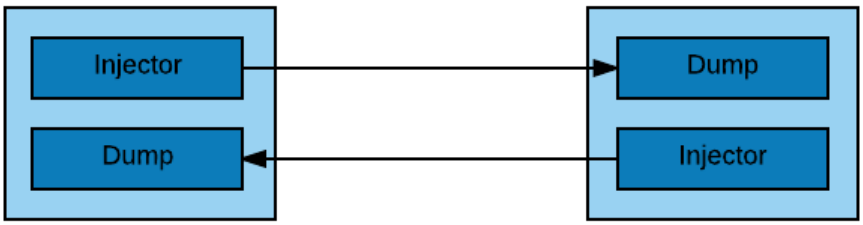


**Ilustración 3 - Conexión de módulo de aplicación**

### 6.3.1 Paquete

El paquete utilizado en esta capa será denominado Aplication. Estos son los parámetros de la cabecera.

- **Secuencia**: número de secuencia del paquete enviado
- **Tam**: Tamaño del paquete enviado

- **interTime**: tiempo entre este paquete y el anterior.

Se utilizará la abreviatura AP para hacer referencia a este tipo de paquetes.

### 6.3.2 Injector

Este módulo tiene como objetivo la generación de los datos que se trasmitirán.

- Parámetros
  - **Dst_Addr:** dirección de transporte destino de los paquetes generados.
    - Defecto: 000
  - **n_Pkt**: número de paquetes a generar. En caso de ser negativo no se establece límite
    - Defecto: -1
  - **delayTime:** tiempo entre paquetes, se deberá introducir en segundos. Permite introducir distribuciones
    - Defecto: 30s
  - **pktSize**: tamaño de los paquetes generados, se deberá introducir en bits. Permite introducir distribuciones
    - Defecto: 40bits
- Puertas
  - **down_out**: puerta de salida hacia la capa inferior. Debe estar conectada
    - **Tipo**: M
    - **Conectada**: obligatorio
- Señales
  - **pktTam**: tamaño del paquete generado en el momento de la generación.

Se considera que el funcionamiento es simple por lo que no es necesaria una exhaustiva descripción del mismo.

### 6.3.3 Dump

Este módulo tiene como objetivo la recepción de paquetes y generación de estadísticas.

- Parámetros
  - *Este módulo no tiene parámetros
- Puertas
  - **down_in**: puerta de entrada para los paquetes de datos recibidos.
    - **Tipo:** M
    - **Conectada**: obligatorio
- Señales
  - **rcvBit**: cantidad de bits recibidos
  - **rcvPkt:** cantidad de paquetes recibidos
  - **bitThroughput**: Throughput en función de los bits recibidos. Se calcula cada vez que se re recibe un paquete
  - **pktThroughput**: Throughput en función de los paquetes recibidos. Se calcula cada vez que se recibe un paquete.

Se considera que el funcionamiento es simple por lo que no es necesaria una exhaustiva descripción del mismo.

## 6.4 Transporte

Esta capa tiene como objetivo la gestión de flujo. A continuación se analizarán el paquete utilizado y los módulos de esta capa.

- **Free_sw**: módulo de gestión de flujo según el protocolo Stop & Wait.
- **Free_gbn:** módulo de gestión de flujo según el protocolo Go Back N.

### 6.4.1 Paquete

El paquete utilizado en esta capa será denominado Transport. A continuación se muestran los datos incluidos en el mismo.

- **seq**: número de secuencia del paquete

- **type**: tipo del paquete, puede ser mensaje o ack
- **srcAddr**: dirección origen
- **dstAddr**: dirección destino

Se utilizará la abreviatura TP para referirse a este tipo de paquetes.

### 6.4.2 free_sw

Este módulo realizará el control de flujo según el protocolo Stop & Wait, también responderá con el ACK correspondiente a los paquetes del tipo mensaje recibidos.

- Parámetros:
  - **Addr**: dirección del módulo de transporte
    - Defecto: 100
  - **Header_Tam**: tamaño de la cabecera de los paquetes de datos
    - Defecto: 0
  - **Ack_Tam:** tamaño de los ACK generados
    - Defecto: 1
  - **Time_Out:** tiempo de espera máximo en segundos para recibir el ACK
    - Defecto: 300
  - **Queue_Tam**: tamaño máximo de la cola. Negativo en caso de no querer límite de cola
    - Defecto: -1
- Puertas
  - **down_out:** conexión saliente hacia la capa inferior
    - **Tipo**: M
    - **Conectada:** obligatoria
  - **down_in**: conexión entrante desde la capa inferior
    - **Tipo:** M

        - **Conectada**: obligatoria
    - **up_out**: conexión saliente hacia la capa superior
        - **Tipo**: M
        - **Conectada**: opcional
    - **up_in:** conexión entrante desde la capa superior
        - **Tipo:** M
        - **Conectada**: opcional
- Señales
    - **rcvACK**: cantidad de ACK recibidos
    - **sndACK:** cantidad de ACK enviados
    - **queueTam:** tamaño de la cola en cada momento.

#### 6.4.2.1 Funcionamiento

A continuación se muestra un diagrama de flujo que indica el comportamiento del módulo al recibir un nuevo mensaje. Se definen dos estados:

- **Idle:** inactividad
- **W_ack**: esperando el ack del mensaje enviado

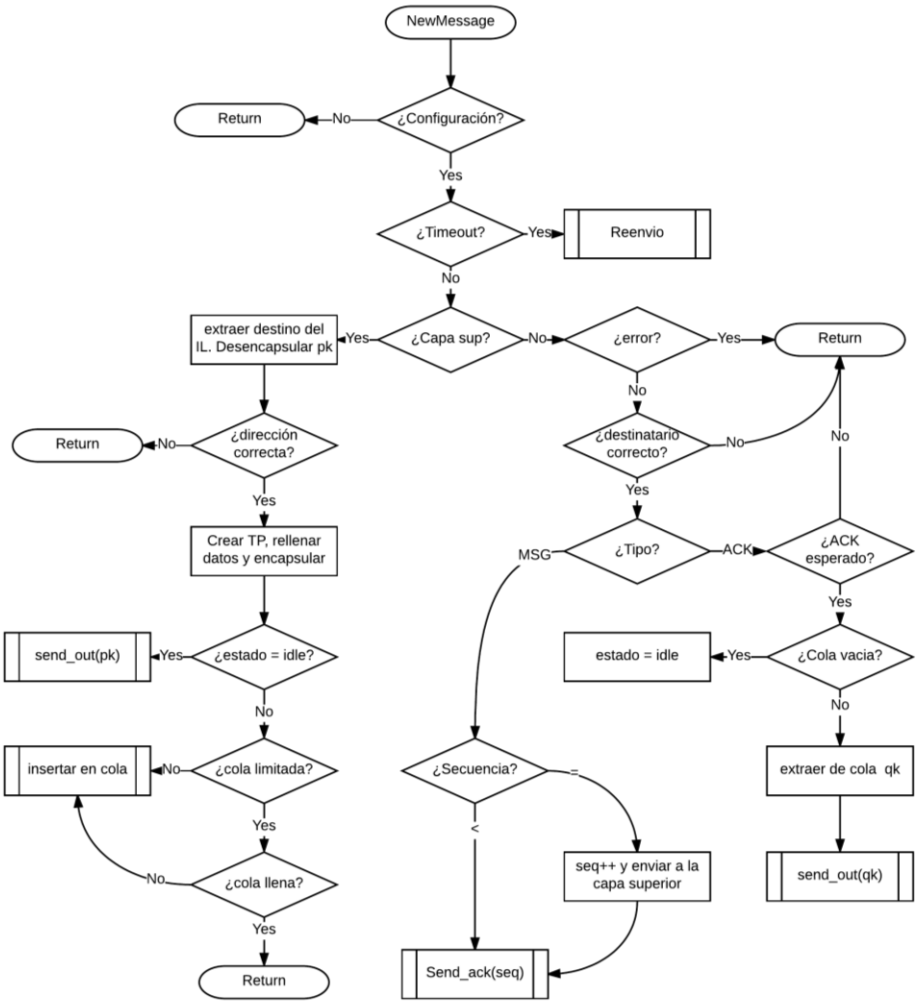


**Ilustración 4 - Diagrama de funcionamiento de free_sw**

### 6.4.3 free_gbn

Este módulo realizará el control de flujo según el protocolo Go Back N, también responderá con el ACK correspondiente a los paquetes del tipo mensaje recibidos.

- Parámetro
  - **Addr**: dirección de transporte del módulo
    - Defecto: 100
  - **Header_Tam:** tamaño de la cabecera de los mensajes de datos
    - Defecto: 0
  - **Ack_Tam:** tamaño de los ACK generados
    - Defecto: 1

- **Queue_Tam:** tamaño de la cola, en caso de ser negativo la cola será infinita
  - Defecto: -1
- **Window_Tam:** tamaño de la ventana de transmisión. Deberá ser positivo
  - Defecto 3
- **Time_Out:** tiempo máximo de espera para el ACK desde el último ACK o el paquete más antiguo transmitido
  - Defecto: 300

- Puerta
  - **down_out:** conexión saliente hacia la capa inferior
    - **Tipo**: M
    - **Conectada:** obligatoria
  - **down_in**: conexión entrante desde la capa inferior
    - **Tipo:** M
    - **Conectada**: obligatoria
  - **up_out**: conexión saliente hacia la capa superior
    - **Tipo**: M
    - **Conectada**: opcional
  - **up_in:** conexión entrante desde la capa superior
    - **Tipo:** M
    - **Conectada**: opcional

- Señales
  - **rcvACK**: cantidad de ACK recibidos
  - **sndACK:** cantidad de ACK enviados
  - **queueTam:** tamaño de la cola en cada momento.

- **sndWindow:** tamaño de la ventana de transmisión en cada momento

#### 6.4.3.1 Funcionamiento

A continuación se muestra un diagrama de flujo que indica el comportamiento del módulo al recibir un nuevo mensaje. Se definen dos estados:

- **Idle:** inactividad
- **Full_w:** ventana llena.

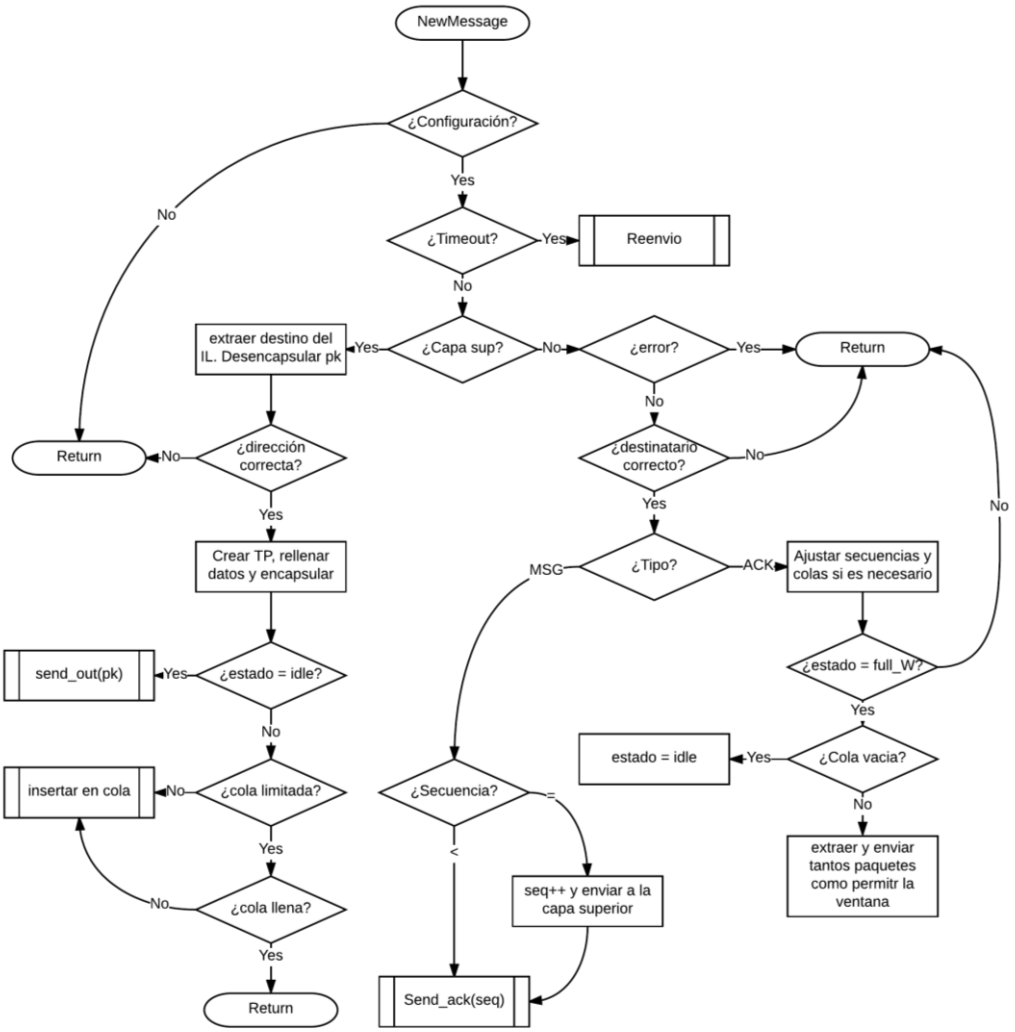


**Ilustración 5 - Diagrama de funcionamiento de free_gbn**

## 6.5 Red

Esta capa tiene como objetivo el rutado de los paquetes recibidos. En este caso, se considera que las diversas alternativas pueden ser correctamente ofrecidas por un solo módulos

### 6.5.1 Paquete

El paquete utilizado en esta capa será denominado Network. A continuación se muestran los datos incluidos en el mismo.

- **srcAddr**: dirección origen
- **dstAddr**: dirección destino
- **hopCount**: cuento de los saltos realizados por el paquete
- **hopLimit**: límite de saltos establecidos para el paquete
- **protocol**: identificador de capa encapsulada

Se utilizará la abreviatura TP para referirse a este tipo de paquetes.

### 6.5.2 Router

Este módulo realiza tareas de rutado según la configuración recibida

- Parámetros
  - **Addr:** dirección de red del módulo
    - Defecto: 00
  - **Hop_Limit**: límite que se impondrá a los paquete generados
    - Defecto: 10
  - **Header_Tam**: tamaño de cabecera de los paquetes generados
    - Defecto: 0
  - **Config_File**: fichero XML de configuración
    - Defecto: fichero de ejemplo
- Puertas
  - **up_out[10]:** array de conexiones saliente hacia la capa superior

- **Tipo:** M
- **Conectada:** opcional

- **up_in[10]:** array de conexiones entrantes desde la capa superior
  - **Tipo:** M
  - **Conectada**: opcional
- **down_out[10]:** array de conexiones saliente hacia la capa inferior
  - **Tipo:** M
  - **Conectada**: opcional
- **down_in[10]:** array de conexiones entrantes desde la capa inferior
  - **Tipo:** M
  - **Conectada:** opcional
- Señales
  - **errorPkt:** cantidad de paquetes erróneos recibidos
  - **rcvBit**: cantidad de bits enviados a la capa superior
  - **rcvPkt:** cantidad de paquetes enviados a la capa superior
  - **forwardedBit**: cantidad de bits rutados
  - **forwardedPkt**: cantidad de paquetes rutados

#### 6.5.2.1 Fichero de configuración

En este apartado se analizará el fichero XML de configuración para el módulo router y un ejemplo del mismo.

- **Routes** (etiqueta)
  - **Route** (etiqueta)
    - **Start** (atributo): dirección destino comienzo del rango
    - **Stop** (atributo); dirección destino final del rango
    - **Gateway** (etiqueta)

- **Id** (atributo): id de la puerta de salida de la capa inferior
- **Prob** (atributo): probabilidad de salida por dicha puerta

```
<?xml version="0" encoding="UTF-8" ?>
<!DOCTYPE routes SYSTEM "route.dtd">

<routes>
        <route start="0" stop="49">
                <gateway id="2" prob="0.5"/>
                <gateway id="1" prob="0.5"/>
        </route>
        <route start="50" stop="99">
                <gateway id="3" prob="1"/>
        </route>
</routes>
```

**Ilustración 6 - Ejemplo de configuración router**

En este ejemplo, los paquetes con destino entre 0 y 49 saldrán por la las puertas 1 o 2 con la misma probabilidad, por el contrario, todos los paquetes con destinos entre 50 y 99 saldrán por la puerta 3.

#### 6.5.2.2 Funcionamiento

A continuación se analizará el comportamiento del módulo al recibir un paquete

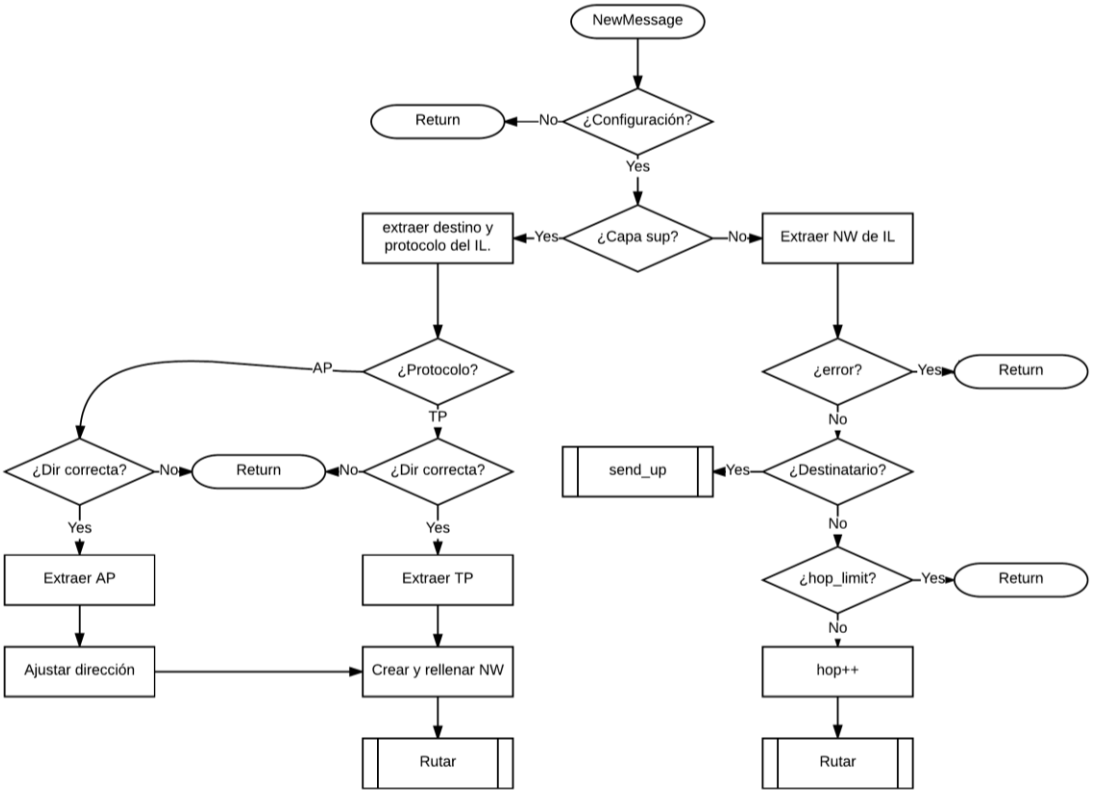


**Ilustración 7 - Diagrama de funcionamiento de router**

## 6.6 Enlace

La capa de enlace tiene como objetivo gestionar el envío de paquetes en el nivel físico. En este apartado se analizará el paquete asociado a dicha capa y los diferentes módulos diseñados. Todos los módulos de esta capa hacen uso de conexiones punto a punto por lo que no es necesario direccionamiento.

Los módulos de la capa se divide en dos clasificaciones: simple gestión de enlace y control de flujo.

- Gestión del enlace
  - Físico
  - Simple_fisico
- Gestión de flujo
  - SenderSW
  - SenderGBN
  - ReceiverACK

En el caso de la gestión de enlace, la comunicación es full dúplex y la conexión deberá ser siempre entre módulos del mismo tipo. En el caso de gestión de flujo la comunicación será unidireccional, el protocolo lo establece el emisor. Se supone que los ACK o NACK de la capa de enlace no pueden ser erróneos.

### 6.6.1 Paquete

El paquete utilizado en esta capa será denominado Link. A continuación se muestran los datos incluidos en el mismo.

- **seq:** secuencia de los paquetes
- **type:** tipo del paquete

Se utilizará la abreviatura LK para referirse a este tipo de paquetes.

### 6.6.2 Fisico

El módulo realizará la gestión física del enlace y control de errores.

- Parámetros
  - **Queue_Length:** tamaño de la cola. En caso de ser un valor negativo será infinita
    - Defecto: -1
  - **Header_Tam:** tamaño de la cabecera
    - Defecto: 0
- Puertas
  - **Out**: conexión saliente hacia otro equipo
    - **Tipo**: C
    - **Conectada**: opcional
  - **In**: conexión entrante desde otro equipo
    - **Tipo:** C
    - **Conectada:** opcional
  - **up_out:** conexión saliente hacia la capa superior
    - **Tipo:** M

        - **Conectada:** opcional

    - **up_in:** conexión entrante desde la capa superior

        - **Tipo:** M

        - **Conectada:** opcional

- Señales

    - **rcvBit:** cantidad de bits correctos recibidos, los que se reenvían a la capa superior

    - **rcvPkt**: cantidad de paquetes correctos recibidos, los que se reenvían a la capa superior

    - **sndBit:** cantidad de bits enviados

    - **sndPkt:** cantidad de paquetes enviados

    - **lostPkt**: cantidad de paquetes eliminados porque la cola estaba llena

    - **errorPkt:** cantidad de paquetes erróneos recibidos

    - **QueueState:** estado de la cola en cada momento

#### 6.6.2.1 Funcionamiento

A continuación se analiza el funcionamiento del módulo al recibir un paquete: La comprobación de cola solo se realizará si el parámetro es positivo.

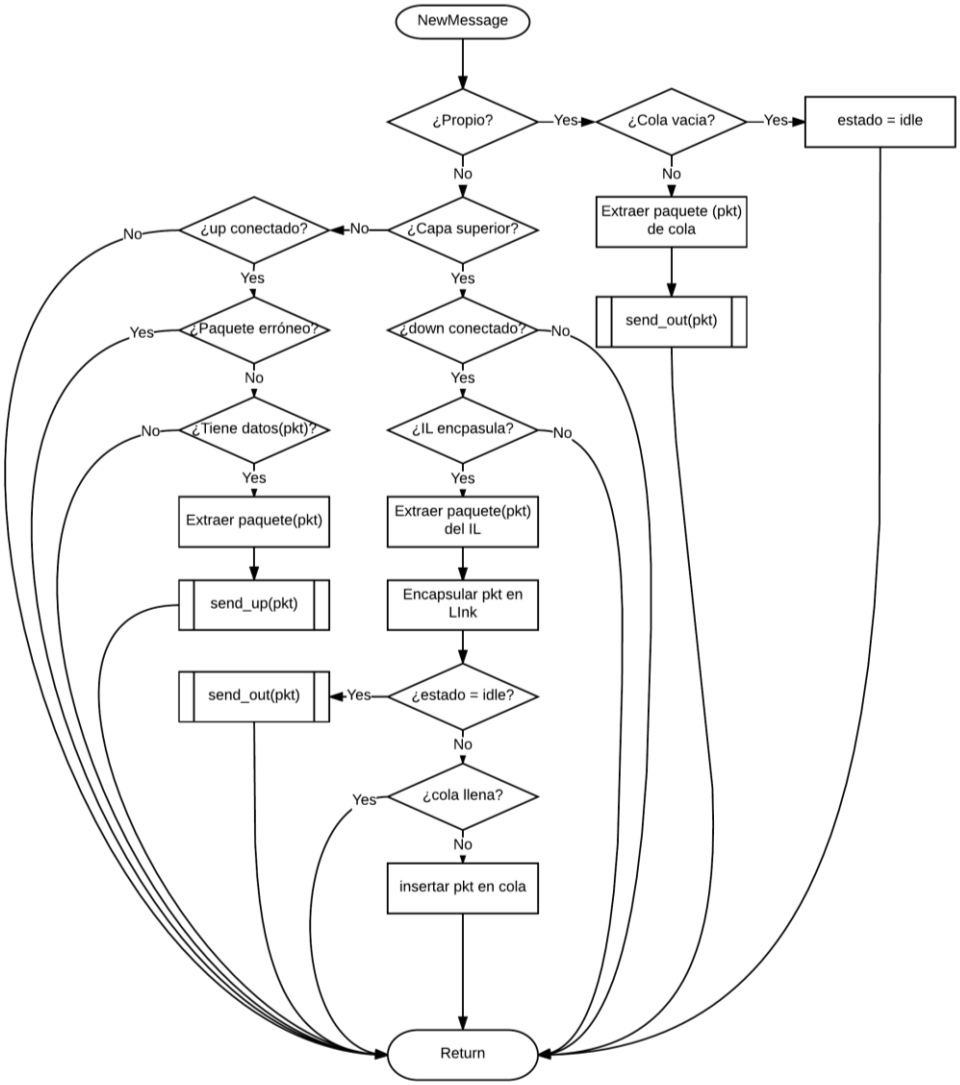


**Ilustración 8 - Diagrama de funcionamiento de fisico**

### 6.6.3 simple_fisico

Este módulo solo realizará la gestión física del enlace, no empaquetará ni comprobará errores.

- Parámetros
  - *Este módulo no tiene parámetros
- Puertas
  - **Out**: conexión saliente hacia otro equipo
    - **Tipo**: C
    - **Conectada**: opcional

- **In**: conexión entrante desde otro equipo
  - **Tipo:** C
  - **Conectada:** opcional
- **up_out:** conexión saliente hacia la capa superior
  - **Tipo:** M
  - **Conectada:** opcional
- **up_in:** conexión entrante desde la capa superior
  - **Tipo:** M
  - **Conectada:** opcional

- Señales
  - **rcvBit:** cantidad de bits correctos recibidos, los que se reenvían a la capa superior
  - **rcvPkt**: cantidad de paquetes correctos recibidos, los que se reenvían a la capa superior
  - **sndBit:** cantidad de bits enviados
  - **sndPkt:** cantidad de paquetes enviados

#### 6.6.3.1 Funcionamiento

A continuación se mostrará el funcionamiento del modelo cuando se recibe un paquete

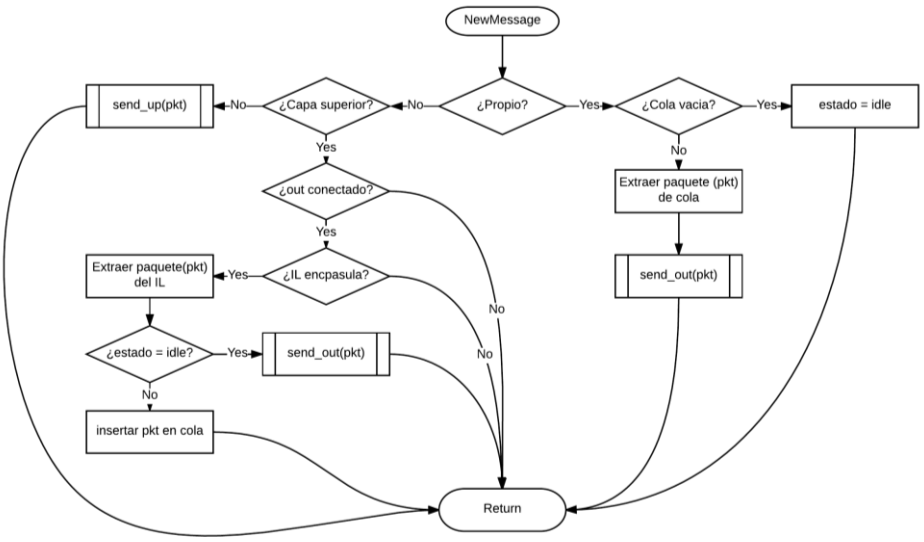


**Ilustración 9 - Diagrama de funcionamiento de simple_fisico**

### 6.6.4 SenderSW

Este módulo enviará los paquetes recibidos de la capa superior según el protocolo Stop & Wait gestionando que el envío se haga correctamente en el canal físico. El módulo no recibirá paquetes, sólo espera recibir ACK del receptor.

- Parámetros
  - **Header_Tam**: tamaño de la cabecera
    - Defecto: 0
  - **Queue_Length:** longitud de la cola. Será infinito si es negativo
    - Defecto: -1
- Puertas
  - **up_in.** puerta de recepción desde la capa superior
    - **Tipo:** M
    - **Conectada**: obligatoria
  - **in**: puerta de entrada de los ACK o NACK del receptor
    - **Tipo**: C
    - **Conectada**: obligatoria

- **out:** puerta de salida de los paquetes de datos
  - **Tipo:** C
  - **Conectada**: obligatoria

- Señales
  - **rcvACK:** cantidad de ACK recibidos
  - **rcvNACK**: cantidad de NACK recibidos
  - **QueueState**: estado de la cola en cada momento
  - **sndBit:** cantidad de bits de datos enviados
  - **sndPkt:** cantidad de paquetes de datos enviados
  - **lostPkt:** cantidad de paquetes perdidos por la cola llena

#### 6.6.4.1 Funcionamiento

A continuación se mostrará la máquina de estados definida para este módulo. Los estados definidos son los siguientes.

- **idle:** sin actividad
- **Sending**: enviando
- **W_ack:** esperando el ACK

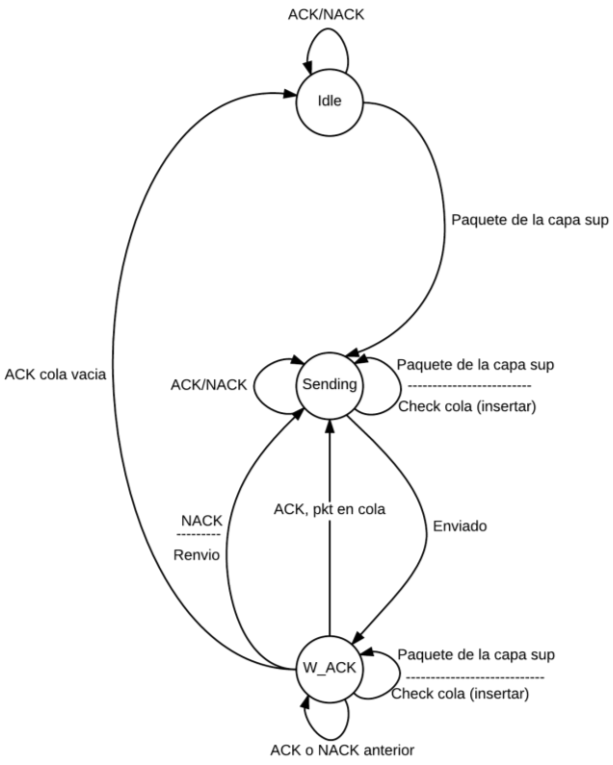


**Ilustración 10 - Maquina de estados de SenderSW**

### 6.6.5 SenderGBN

Este módulo enviará los paquetes recibidos de la capa superior según el protocolo Go Back N gestionando que el envío se haga correctamente en el canal físico. El módulo no recibirá paquetes, sólo espera recibir ACK del receptor.

- Parámetros
  - **Header_Tam**: tamaño de la cabecera
    - Defecto: 0
  - **Queue_Length:** longitud de la cola. Será infinito si es negativo
    - Defecto: -1
- Puertas
  - **up_in.** puerta de recepción desde la capa superior
    - **Tipo:** M
    - **Conectada**: obligatoria
  - **in**: puerta de entrada de los ACK o NACK del receptor

        - **Tipo**: C
        - **Conectada**: obligatoria
    - **out:** puerta de salida de los paquetes de datos
        - **Tipo:** C
        - **Conectada**: obligatoria

- Señales
    - **rcvACK:** cantidad de ACK recibidos
    - **rcvNACK**: cantidad de NACK recibidos
    - **QueueState**: estado de la cola en cada momento
    - **WindowTam**: estado de la ventana
    - **sndBit:** cantidad de bits de datos enviados
    - **sndPkt:** cantidad de paquetes de datos enviados
    - **lostPkt:** cantidad de paquetes perdidos por la cola llena

#### 6.6.5.1 Funcionamiento

A continuación se analizará la máquina de estados de este módulo. Los posibles estados son los siguientes:

- **Idle**: inactividad
- **Sending_i**: enviando paquetes de la cola de transmisión
- **Sending_R:** reenviando los paquetes de la ventana

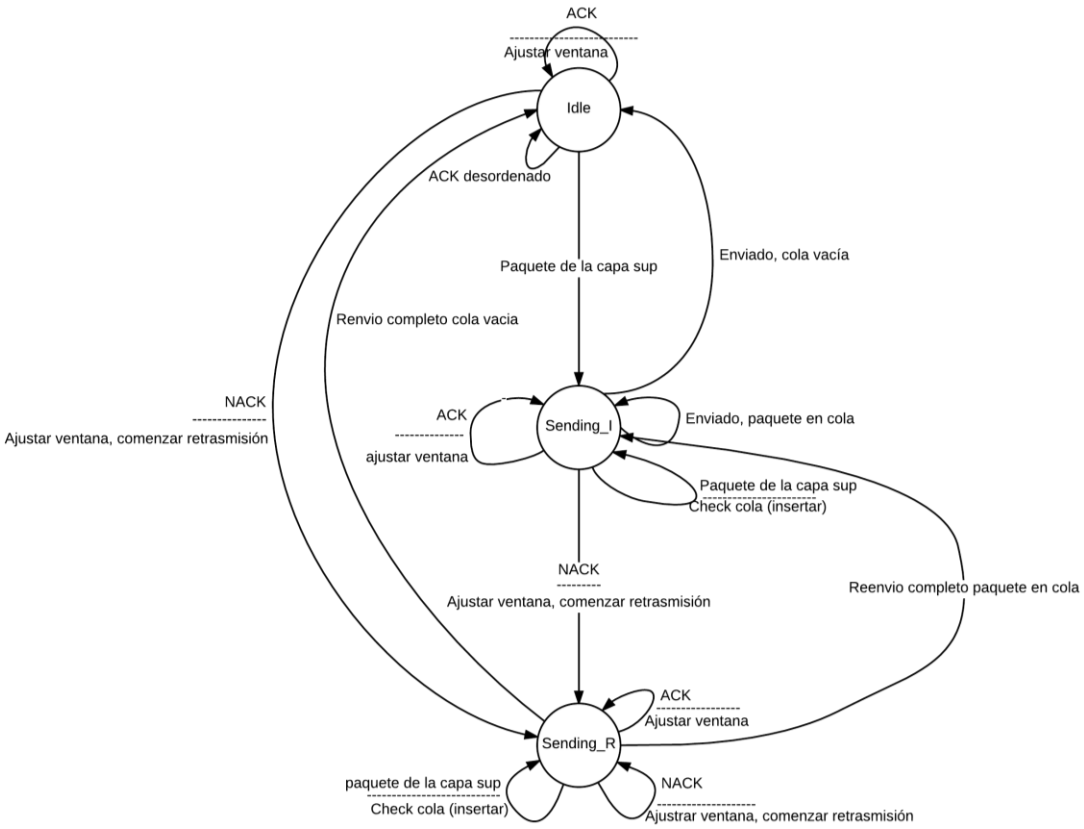


**Ilustración 11 - Máquina de estados de SenderGBN**

### 6.6.6 ReceiverACK

Este módulo es el receptor de los módulos anteriores, SenderSW y SenderGBN. El objetivo de este módulo es recibir paquetes de datos, responder con el ACK o NACK correspondiente y reenviar los correctos a la capa superior.

- Parámetros
  - **Ack_Tam:** tamaño de los ACK o NACK generados
    - Defecto: -1
- Puertas
  - **in**: puerta de entrada de los paquetes de datos
    - **Tipo**: C
    - **Conectada**: obligatoria
  - **out:** puerta de salida de los ACK o NACK hacia el emisor

        - **Tipo:** C
        - **Conectada**: obligatoria
    - **up_out:** puerta de salida hacia la capa superior
        - **Tipo**: M
        - **Conectada**: Obligatoria

- Señales
    - **sndACK:** ACK enviados
    - **sndNACK:** NACK enviados
    - **rcvBit:** bits correctos recibidos
    - **rcvPkt:** paquetes corrector recibidos

#### 6.6.6.1 Funcionamiento

A continuación se comprobará el funcionamiento del módulo al recibir un paquete.

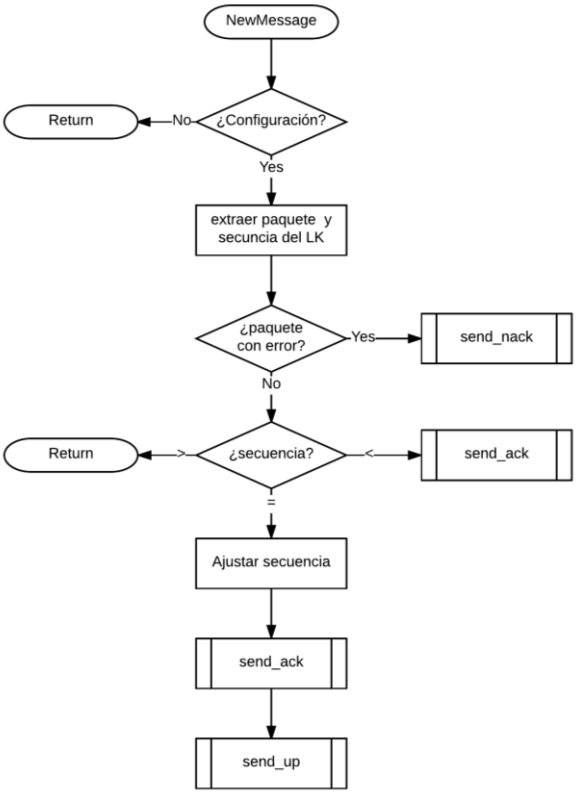


**Ilustración 12 - Diagrama de funcionamiento de ReceiverACK**

## 6.7 Multiplexación

Aunque la multiplexación no es considerada una capa como tal pero los módulos generados con este fin se tratarán por separado ya que se comunican directamente entre sí con paquetes específicos como el resto de capas.

Estos son los módulos que se generarán:

- Multiplexores
    - Multiplexor simple
    - Multiplexor con prioridad
- Demultiplexor

La imagen a continuación muestra cómo se realizará la conexión entre los mismos

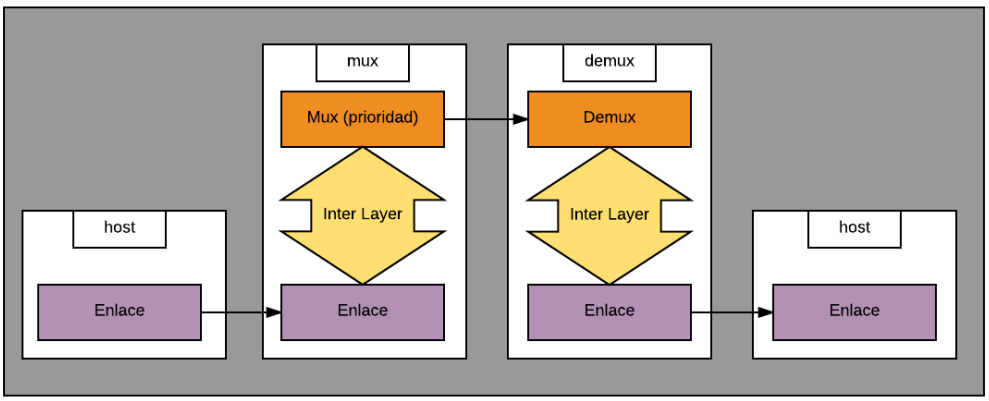


**Ilustración 13 - Estructura de multiplexación**

La multiplexación se organizará en líneas para que el demultiplexor pueda identificar la salida para el paquete recibido. Los datos multiplexados irán encapsulados en paquetes de multiplexación que incluirán los datos necesarios para completar las funciones de multiplexación y demultiplexación.

Cada entrada tendrá asociada una línea y una prioridad en caso de ser el multiplexor con prioridad. Cada línea tendrá asociada una salida en el demultiplexor.

### 6.7.1 Paquete

El paquete utilizado en esta capa será denominado Mux. Estos son los parámetros de la cabecera.

- **Linea**: identificador de la línea del paquete

Se utilizará la abreviatura MX para hacer referencia a este tipo de paquetes.

### 6.7.2 Mux

Este módulo tiene como objetivo multiplexar en un solo canal los paquetes recibidas desde las distintas entradas.

- Parámetros
  - **config:** fichero XML de configuración
    - Defecto: fichero de ejemplo
  - **Header_Tam:** tamaño de la cabecera de multiplexación
    - Defecto: 0
- Puertas
  - **out**: salida hacia el canal de multiplexación
    - **Tipo:** C
    - **Conectada:** obligatorio
  - **in[]:** array de conexiones de entrada
    - **Tipo:** M
    - **Conectada:** opcional
- Señales
  - *Este módulo no genera señales

#### 6.7.2.1 Fichero de configuración

El fichero de configuración de se analizará junto con el multiplexor con prioridad ya que es el mismo.

#### 6.7.2.2 Funcionamiento

A continuación se mostrará el funcionamiento del módulo al recibir un paquete. Se establecen dos posibles estados:

- **Idle:** inactivo
- **Sending:** enviando un paquete por el canal de multiplexación

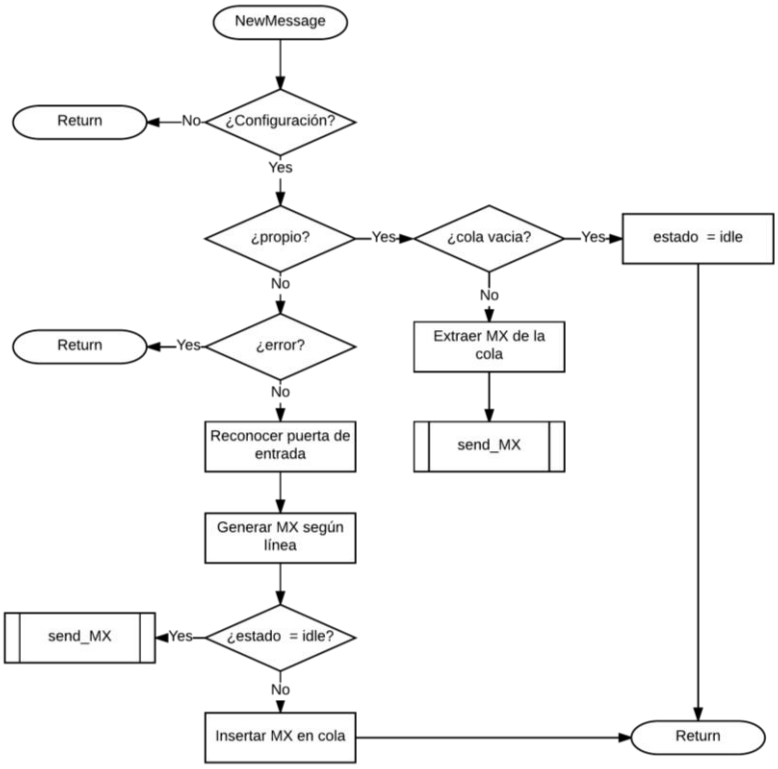


**Ilustración 14 - Diagrama de funcionamiento de mux**

### 6.7.3 Mux_priority

Este módulo tiene como objetivo multiplexar los paquetes recibidos desde las diferentes entradas en un único canal de multiplexación según las prioridades y líneas configuradas.

- Parámetro
  - **Config:** fichero XML de configuración
    - Defecto: fichero de ejemplo
  - **Header_Tam:** tamaño de la cabecera de multiplexación.
    - Defecto: 0
- Puertas
  - **out**: salida hacia el canal de multiplexación
    - **Tipo:** C
    - **Conectada:** obligatorio
  - **in[]:** array de conexiones de entrada

- **Tipo:** M
- **Conectada:** opcional

- Señales
  - *Este módulo no genera señales

#### 6.7.3.1 Fichero de configuración

A continuación se mostrará la estructura del fichero y un ejemplo. El atributo de prioridad será obviado por el módulo de multiplexación simple.

- **Inputs** (etiqueta)
  - **Input** (etiqueta)
    - **Gate** (atributo): puerta de entrada
    - **Line** (atributo): línea a la que se asociarán los paquetes de la puerta
    - **Priority** (atributo): prioridad asociada a los paquetes de la puerta

```
<?xml version="0" encoding="UTF-8" ?>
<!DOCTYPE routes SYSTEM "mux.dtd">

<inputs>
        <input gate="0" line="0" priority="0"/>
        <input gate="1" line="1" priority="10"/>
        <input gate="2" line="2" priority="10"/>
</inputs>
```

**Ilustración 15 - Fichero de configuración de mux**

En este caso cada entrda tendrá el mismo número de línea, la entrada 0 tiene prioridad 0, mientras que las entradas 1 y 2 tienen prioridad 10. Los paquetes de las entradas 1 y 2 compartirán cola mientras que los paquetes de la entrada 0 sólo se transmitirán si esa primera cola está vacía.

#### 6.7.3.2 Funcionamiento

A continuación se analizará el funcionamiento cuando llega un nuevo paquete a multiplexar. Se definen dos posibles estados:

- **Idle:** inactivo
- **Sending:** Enviando

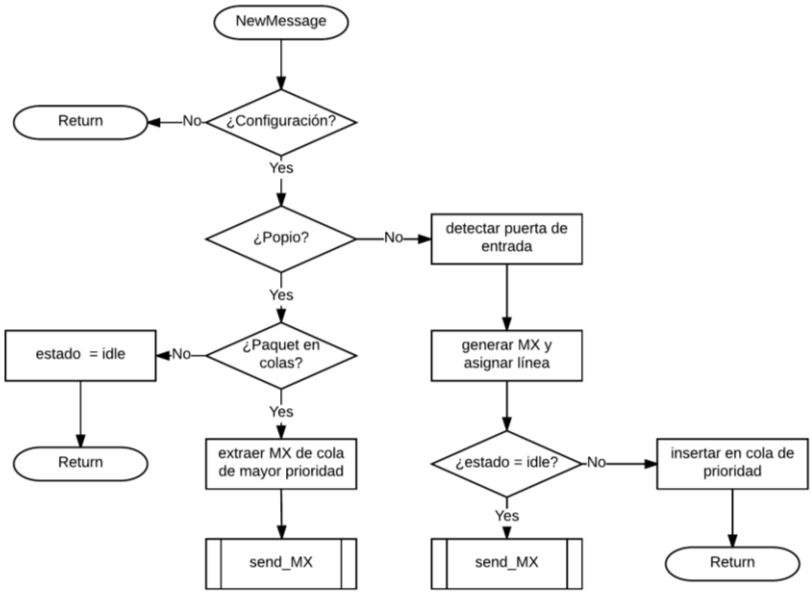


**Ilustración 16 - Diagrama de funcionamiento de mux_priority**

### 6.7.4 Demux

Este módulo demultiplexa los paquetes en el canal de multiplexación en las distintas salidas asociadas al mismo. Las líneas de los paquetes indican por qué salida se debe reenviar el paquete multiplexado.

- Parámetros
  - Config: fichero XML de configuración
    - Defecto: fichero de ejemplo
- Puertas
  - **Out[]:** array de salidas hacia la capa inferior
    - **Tipo:** M
    - **Conectada:** opcional
  - **In**: puerta de entrada desde el canal de multiplexación
    - **Tipo**: C
    - **Conectada**: obligatorio

- Señal
    - **rcvBit**: bits demultiplexados
    - **rcvPkt:** paquetes demultiplexados.

#### 6.7.4.1 Fichero de configuración

A continuación se mostrará la estructura del fichero y un ejemplo.

- **Lines** (etiqueta)
    - **Line** (etiqueta)
        - **Line** (atributo): línea identificativa de los paquetes recibidos
        - **Gate** (atributo): puerta de salida de los paquetes recibidos

```
<?xml version="0" encoding="UTF-8" ?>
<!DOCTYPE routes SYSTEM "demux.dtd">

<lines>
        <line line="0" gate="0" />
        <line line="1" gate="1" />
        <line line="2" gate="2" />
</lines>
```

**Ilustración 17 - Fichero de configuración de demux**

En este caso cada línea identifica al mismo tiempo la salida por la que se retrasmitirán los paquetes recibidos.

#### 6.7.4.2 Funcionamiento

En este apartado se analizará el funcionamiento del módulo al recibir un paquete desde el canal de multiplexación.

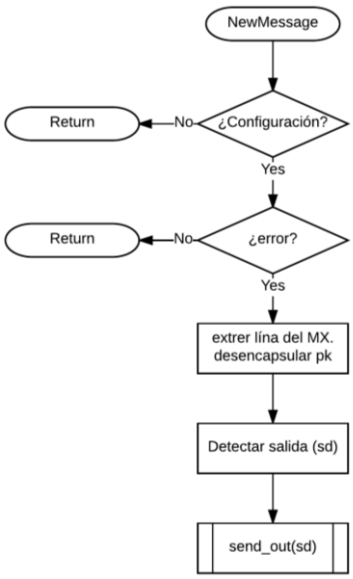


**Ilustración 18 - Diagrama de funcionamiento de demux**