UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Simulación y estudio del rendimiento de redes p2p

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Sánchez Cuevas, Alejandro

Tutor/a: Oliver Gil, José Salvador

CURSO ACADÉMICO: 2021/2022

# Resumen

El presente trabajo Final de Grado se centrará en el estudio de las redes Peer to Peer (P2P) y las diferentes arquitecturas que estas pueden adquirir y su posterior simulación

Por ello, el objetivo principal que este trabajo sostiene es el estudio teórico de las redes P2P y las diferentes arquitecturas que estas pueden establecer y la posterior simulación de la lógica de su funcionamiento. Para ello se hará uso del entorno de simulación OMNeT++, con el fin de estudiar la eficiencia de estas.

Para poder analizar su eficiencia, las redes definidas serán sometidas a una serie de pruebas en las que se tendrá en cuenta una serie de parámetros como el número de nodos que formen la red, la velocidad de transmisión de los canales, etc.

Para finalizar se realizará una comparativa con los resultados obtenidos para determinar la mejor topología para cada red.

**Palabras clave:** Peer to Peer, OMNeT++, entorno de simulación, arquitecturas, eficiencia

# Abstract

The present Final Degree Project focuses on the study of Peer to Peer networks and their different architecture and their subsequent simulation.

Therefore, the main goal that this Project hold is the theoretical study of P2P networks and their different architectures and his following simulation of the logical functioning, for that purpose, the simulation Environment OMNeT++ will be used, to study the efficiency of this.

To analyze its efficiency, the defined networks will be subdued to some tests, where a list of parameters like, the number of nodes that form the network, channels transmission speed, etc.

To conclude, a comparison between the obtained results will be made, to decide which topology for each architecture is better.

**Keywords:** Peer to Peer, OMNeT++, simulation Environment, architectures, eficiency

# Tabla de contenidos

# Tabla de figuras

# 1. Introducción

Las redes *Peer to Peer* (por sus siglas en inglés, P2P) son un tipo de red que, a diferencia de las redes Cliente-Servidor, no cuentan con un servidor central que trate las peticiones de los clientes, sino que los propios nodos de la red hacen tanto de cliente como de servidor compartiendo sus recursos hardware para proveer a la red de los recursos y servicios necesarios [R. Schollmeier, 2001].

Considerando que, en la actualidad, en las redes hay conectadas una cantidad abrumadora de dispositivos y, viendo la problemática que tienen las redes Cliente-Servidor (estas requieren de un servidor central para su funcionamiento), sería recomendable la realización de un estudio de comportamiento de las redes P2P como alternativa a las redes anteriormente mencionadas.

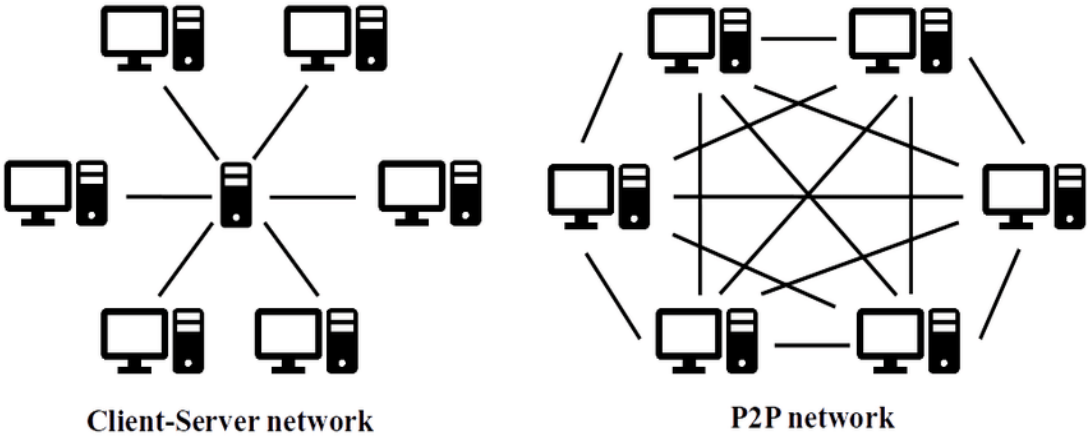


*Figura 1 Diagrama redes Cliente-Servidor vs redes Peer to Peer*

## 1.1 Arquitecturas existentes

Las Redes Cliente-Servidor son un tipo de redes centralizadas en la que hay dos tipos de componentes principales, uno de ellos son los clientes y los segundos son los Servidores [Shakirat, 2014]. Los clientes se conectan a esta red, realizando peticiones a los Servidores, y son los Servidores los que están a la espera de cualquier petición que les hagan los clientes. Estos, actúan como dispositivos que proporcionan servicios necesitados por los clientes, siendo estas peticiones cualquier cosa, desde un ping para comprobar la conexión, como la solicitud de un fichero, por ejemplo [Gilbert Herd, 2000].

Dentro de las redes Cliente-Servidor los componentes se pueden dividir en dos tipos: componentes físicos y componentes lógicos.

Entre los componentes físicos dentro de una red muy básica de tipo cliente-servidor se encuentran [Shakirat, 2014].

- **Cliente**: puede ser un programa o un proceso que se encargar de enviar las peticiones al servidor a través de la red. Un cliente puede tratarse de un navegador web, un escritorio remoto, etc.
- **Servidor**: es el encargado de escuchar las peticiones que le llegan de los clientes a través de la red. Los servidores suelen estar ejecutándose en *mainframes*.

En cuanto a los componentes lógicos se encuentran [Shakirat, 2014].

- **Lenguaje de Programación**: es la forma en la que se pueden comunicar el cliente y el servidor, este lenguaje puede ser vía scripts (js, phyton, etc.), vía HTML, etc.
- **Protocolos de Comunicación**: son los protocolos que se pueden seguir para establecer la conexión y comunicarse entre cliente y servidor. Estos protocolos pueden ser http, https, SSL, etc.

En la actualidad las redes Cliente-Servidor son el tipo de red más usada debido a su característica centralización de los servidores que atienden las peticiones a los clientes, permite un mayor control de los datos, una mayor seguridad de la red y un mayor control del flujo de la red; ya que se podría observar las peticiones de los clientes a los servidores y las respuestas de estos. Esta mecánica de trabajo permite evitar, por ejemplo, la propagación de virus. [Peter Duchessi, InduShobha Chengalur-Smith, 1998].

Una de las principales problemáticas de las redes Cliente-Servidor es también una de sus ventajas, que es la centralización de los servidores. Esto permite tener un mayor control de la red, pero, a su vez, provoca su dependencia de los servidores que se encuentran centralizados; en el caso de que un servidor caiga, la red dejará de ser funcional. Una red no funcional desatiende las peticiones mandadas por los

clientes e incumple su funcionalidad [Mishal Roor, 2020]. A su vez las redes Cliente-Servidor, tienen un problema en cuanto la escalabilidad, y es que, según aumenta el número de usuarios que forman la red se tendrá que realizar inversiones en añadir más servidores a la red, porque si no, es posible que la red se sature y ni pueda atender a todas las peticiones de los clientes.

Un ejemplo de esta problemática fue lo ocurrido en Facebook: la caída de uno de los componentes de los servidores centrales (BGP y DNS) provocaron el mal funcionamiento de la red, induciendo a que los clientes no fueran capaces de utilizar todas las aplicaciones de la compañía (Facebook, WhatsApp e Instagram) [Bruno Toledano, 2021].

Debido a esto, las redes P2P se presentan como una alternativa de alto potencial y, por ello, su funcionamiento será descrito en el Capítulo 3.

## 1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado, en adelante TFG, consistirá en la simulación de tres redes P2P con diferentes topologías: en malla, árbol e híbrida. Su estudio permitirá analizar su rendimiento en base a una serie de variables como: latencia, número de usuarios, entre otras.

El objetivo principal será apoyado por una serie de subobjetivos que permitirán su consecución. Esto son:

- Estudiar y comprender la mecánica de trabajo de OMNeT++. Este entorno de trabajo permite la recopilación de datos de forma rápida y sencilla.
- Seleccionar y analizar los resultados proporcionados por OMNeT ++.
- Plasmar la información de forma gráfica para facilitar su lectura y comprensión.
- Emplear los datos obtenidos para realizar una comparación efectiva y eficiente de las redes P2P atendiendo a sus diferentes topologías.

## 1.3 Estructura

Este TFG se ha organizado en 8 capítulos. Esto permite explicar, de forma teórica, las redes a emplear, su entorno de simulación y, a su vez, la implementación de las redes explicadas, previamente en este entorno.

El primer capítulo de introducción sirve como punto de entrada a la compresión de las redes P2P.

En el segundo capítulo se definirá teóricamente las redes P2P, además de las diferentes topologías que esta puede adquirir donde se explicarán el funcionamiento de las seleccionadas como caso de estudio en este TFG.

En el tercer capítulo se contemplará el estado del arte en el que se mostrará el estado actual de las redes P2P y se expondrán diferentes ejemplos de aplicaciones que utilizan este tipo de red,

En el cuarto capítulo se presentará el entorno de simulación escogido para este TFG, a su vez se hablará de las capacidades de este y de las limitaciones que tiene.

En el quinto capítulo se explicará cómo se han implementado las diferentes redes P2P con sus respectivas topologías (malla, árbol e híbrida), además se explicarán los diferentes casos de estudios a los que se pondrán a prueba las redes.

En el sexto capítulo se realizará un estudio de los resultados obtenidos por las distintas simulaciones realizadas.

En el séptimo capitulo se presentarán las conclusiones que se hayan obtenido con los resultados obtenidos en el estudio previo realizado en el capítulo anterior.

Para finalizar, en el octavo capítulo se halla la bibliografía, donde se listan las fuentes de las que se ha obtenido la información para realizar este TFG.

## 1.4 Metodología

La obtención de los objetivos descritos en el Apartado 1.2 es posible gracias al desarrollo de la siguiente metodología:

El primer paso que realizar es un estudio teórico sobre las redes P2P, atendiendo a las ventajas y desventajas que presenta su uso. También, es necesario llevar a cabo una comparativa de dichos factores respecto a las redes de tipo Cliente-Servidor.

A continuación, se realizará un estudio específico sobre las redes objeto de estudio, de tipo P2P, poniendo el foco sobre sus topologías: malla, árbol e híbrida.

Seguidamente, será necesaria la comprensión del entorno de simulación seleccionado: OMNeT ++. Se atenderán sus ventajas y limitaciones.

Para finalizar, las redes simuladas en el entorno de desarrollo permitirán un estudio comparativo de las mismas y un análisis de conclusiones.

# 2. Redes P2P

## 2.1 Introducción

### 2.1.1 ¿Qué es una red cliente-servidor?

Una red cliente-servidor es un tipo de red distribuida, en la que el servidor es la unidad central y además es el único proveedor de contenidos y servicios. Por otra parte, el cliente es un nodo dentro de la red que tan solo hace peticiones de *request* al servidor para obtener los servicios necesarios, sin compartir ninguno de sus propios recursos.

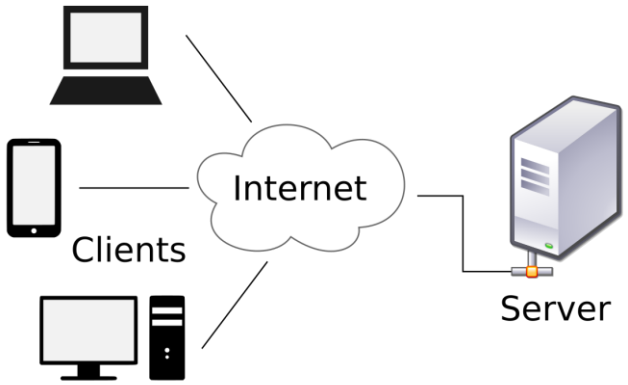


*Figura 2 Red Cliente-Servidor*

La comunicación entre cliente y servidor se basa en envió de mensajes *request-response*, en el que el cliente le envía una petición *request* al servidor y este le contesta con un *response* con la información/servicio solicitado.

El protocolo TCP en la mayor parte de las ocasiones mantiene la comunicación activa entre servidor y cliente hasta que la comunicación entre estos haya finalizado. El protocolo TCP determina la forma óptima de distribuir la información requerida en paquetes que la red puede transportar.

Las redes cliente-servidor se pueden organizar en diferentes arquitecturas, pero debido a que este TFG se centra en las redes P2P tan solo serán explicadas brevemente.

Las posibles arquitecturas que pueden implementar una red cliente-servidor son:

- Arquitectura de un nivel. En este tipo de arquitecturas todas las capas de una aplicación; presentación, lógica, acceso a datos, etc. Están en un mismo dispositivo.
- Arquitectura de dos niveles. En este tipo de arquitectura la aplicación está dividida en dos partes; la parte del cliente, la cual contiene la capa de presentación y la parte del servidor, tiene la capa de acceso a datos.
- Arquitectura de tres niveles. En esta arquitectura la aplicación/servicio está dividido en tres capas. La capa del cliente que contiene la presentación, la capa lógica estaría en un primer servidor y la capa de datos estaría en otro servidor.
- Arquitectura de n niveles. La arquitectura de n niveles es parecida a la arquitectura de tres niveles, pero el número de servidores aumenta distribuyendo más la capa lógica.

### 2.1.2 **¿Qué es una red P2P?**

Las redes Peer to Peer surgen como una alternativa a las redes cliente-servidor, con la finalidad de descentralizar la red, para ello el primer principio de las redes P2P es la eliminación de un servidor central que contenga la información y servicios que se ofrecen y así poder descentralizar la información de la red, aunque más adelante se mencionará un tipo de red P2P con un servidor central.

Por tanto, se puede definir las redes P2P como:

"Una red distribuida puede llamarse red Peer to Peer, si sus participantes comparten entre sí parte de sus recursos hardware (almacenamiento, poder procesamiento, etc.) para poder así establecer los servicios y contenidos ofrecidos por la red. Estos deben de poder ser accesibles por otros nodos de forma inmediata, sin la necesidad de ningún intermediario. Estos nodos que forman la red deben de ser proveedores de servicios a la par que clientes que hacen uso del servicio ofrecido" [R. Schollmeier, 2001].

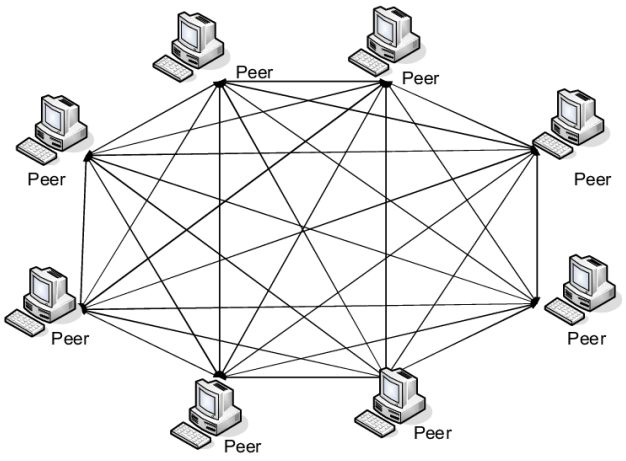


*Figura 3 Red Peer to Peer*

Las redes P2P se distinguen de la computación distribuida tradicional en diversos aspectos. Algunos de ellos son:

**Escalabilidad**. A diferencia de los sistemas distribuidos tradicionales, las redes P2P pueden escalar a miles de nodos. Es por ello, que pueden aprovechar el poder de la computación sobre el internet.

**Heterogeneidad**. Un sistema P2P puede ser heterogéneo en cuanto a la capacidad de hardware de los nodos. Uno puede ser muy lento y otro puede ser muy rápido.

**Dinamismo**. Las aplicaciones P2P suelen trabajar en entornos muy dinámicos. Esta topología puede cambiar muy rápido debido a la unión de nodos nuevos o a la salida de otros.

### 2.1.3 **Beneficios de una red P2P**

Las redes P2P tienen mucho potencial, ya que podría suplir muchas de las necesidades que se tienen tanto a nivel personal como organizativo.

Los beneficios de una red P2P se pueden listar de la siguiente forma.

- Reduce los recursos computacionales sin tener que recurrir a un coste excesivo.
- Permite que la información sea diseminada efectivamente.
- Las redes P2P ejercen un control total sobre los datos que se manejan en su red, y es flexible a la vez con lo que se puede alojar toda la

información en una sola máquina o dividir en fragmentos los datos que se almacena en la red y repartirlos por los distintos nodos.

- En cuanto a la privacidad, se podría publicar información a la red de forma anónima si el publicador se "sumergiera" en grupo de nodos que pasaran a tener la responsabilidad sobre el fichero, borrando así cualquier conexión entre el fichero y el verdadero propietario.

Debido a estos beneficios que se pueden obtener usando una red P2P es por lo que surgieron diversas aplicaciones que hacen uso de estas redes, que más adelante en el Capítulo 4 se expondrán unos ejemplos sobre estas aplicaciones.

### 2.1.4 **Desventajas de una red P2P**

Tal y como las redes P2P tienen beneficios, también tiene una serie de inconvenientes con los que la red tiene que lidiar. Estas desventajas se pueden listar de la siguiente forma.

- En cuanto a la disponibilidad. En el entorno de las redes P2P, los nodos que forman parte de la red son autónomos por lo que se pueden unir o irse de la red en cualquier momento. Esta situación provoca que el esquema de la red sea impredecible. Además, esto provoca que puede ser que en algún momento un recurso no esté disponible en la red cuando es necesitado, es por ello por lo que el mecanismo de replicación de datos puede aliviar este problema.
- En lo que respecta a la ejecución. Una misma petición dentro de una red P2P que sea resuelta en tiempos distintos, o puede que incluso no sea atendida o que devuelva respuestas diferentes. Todo esto depende de la conectividad de los nodos y de la topología de la red que una petición sea atendida o no, y si es atendida cuanto tiempo tarda en realizarse.
- En cuanto a la integridad. En los dos puntos anteriores una de las posibles soluciones para mitigar los problemas es la replicación de los datos, pero esto genera un problema a nivel de integridad en la red. En una red P2P. Como se ha mencionado, los datos pueden ser replicados y

almacenados en diversos nodos, y esto hace que sea muy difícil de mantener una consistencia en las copias de los archivos, ya que si uno se modifica el resto deberían de modificarse, es por ello por lo que la integridad es una cuestión para tener actualmente en las redes P2P.

- En cuanto a la seguridad. Los sistemas P2P presentan agujeros en la seguridad, como cualquier sistema hoy en día. La ideología de P2P, que se basa en compartir públicamente ficheros, hace que sus problemas de seguridad sean más agudos. Permiten a otros nodos acceder al contenido o servicio de otros nodos de la red, lo que hace un nodo sea más vulnerable para atacar en situaciones en las que hace solo de cliente, esto hace que la red sea vulnerable a ataques de denegación de servicios (DoS). Las redes desestructuradas en particular seran las más débiles a sufrir inundaciones de la red mediante peticiones a través de nodos maliciosos.

### 2.1.5 **Redes Cliente-Servidor vs Redes P2P**

Las redes cliente-servidor y las redes P2P tiene diversos factores que las diferencian. Estos factores se pueden listar de la siguiente forma.

- En las redes cliente-servidor, si se quiere una gran escalabilidad, se requiere de dispositivos con muchos recursos, con una alta potencia computacional. Esto requeriría de una gran inversión de dinero, mientras que en las redes cliente servidor los dispositivos no deben de tener una alta potencia computacional.
- Los participantes de las redes-cliente servidor son clientes normales que hacen las peticiones, pero por parte del servidor mayoritariamente son empresas que certifican que ofrecen un servicio seguro y que son confiables, mientras que en las redes P2P, los usuarios de la red que hacen tanto de cliente como servidor, son normalmente usuarios de los que no podemos saber sus intenciones de sobre la red, por lo que en una red P2P es más difícil de gestionar la seguridad de esta.
- Las redes P2P, al ser formada por usuarios tradicionales y no depender de un servidor central que contenga la información, es más probable que

escale más rápidamente y obtenga una magnitud mayor en comparación con una red servidor que tardaría más.

- En comparación con una red P2P, las redes cliente-servidor están mejor estructuradas y son más estables. Debido a esto, es menos probable que los recursos solicitados no estén disponibles en momentos concretos, mientras que, en las redes P2P la estructura es más inestable ya que los nodos pueden unirse o irse cuando quieran, es posible que en momentos concretos parte de la información solicitada no se encuentre disponible.
- Debido a la organización de las redes P2P y de su inestabilidad, las respuestas a las peticiones no son constantes si son respondidas, sino que puede que el tiempo en ser resueltas varie, o que la respuesta sea la incorrecta debido al proceso de comunicación en estas redes, a diferencia de las redes cliente-servidor, donde la respuesta a las peticiones es más constante.

## 2.2 Arquitectura de las redes P2P

Dentro de las redes P2P es posible diferenciar diversas arquitecturas. En general se pueden dividir en 2 categorías distintas. En primer lugar, estaría la arquitectura centrada que se basa, como en las redes cliente-servidor, en el uso de uno o más servidores centrales. Después estarían la arquitectura descentralizada, en la que no hay ningún servidor central como elemento principal de la red. A su vez, dentro de la arquitectura descentralizada es posible dividir está en dos niveles, el primer nivel es si la red es plana (un nivel de profundidad) o si la red es jerárquica (varios niveles de profundidad).

A su vez, hay que destacar que, aparte de estos dos grupos principales, también hay una tercera categoría que es la arquitectura híbrida que combina elementos de la arquitectura centralizada como de la descentralizada, obteniendo las ventajas de ambos.

### 2.2.1 Arquitectura Centralizada

La arquitectura centralizada combina características tanto de las redes cliente-servidor como de las redes P2P. A la par que red cliente-servidor la arquitectura centralizada contiene uno o varios servidores centrales, que ayudan a los otros nodos de la red a saber que nodo contiene la información que ellos buscan. Para

localizar a los nodos que contiene la información cada al inicio de la red le envía un mensaje al servidor indicándole que información tiene él. Pero como toda red P2P el proceso de traspaso de información debe ser descentralizado. Es por ello por lo que una vez el cliente solicita la información al servidor este le da la dirección del nodo que contiene esa información y a partir de ahí la comunicación es entre el nodo solicitante y el nodo que contiene la información aquí el servidor no aporta nada.

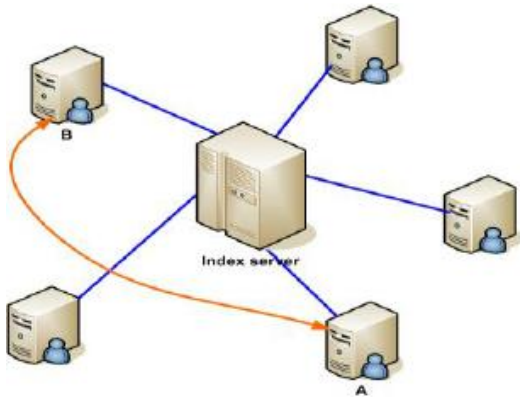


*Figura 4 Red P2P con Arquitectura Centralizada*

Como todo sistema centralizado, este tipo de red P2P es susceptible a sufrir a ataques *single point failure* ya que si el servidor es atacado y este deja de funcionar la red ya no puede funcionar. También hay que recalcar que otro problema dentro de esta arquitectura es que el servidor central puede hacer de cuello de botella al resto de la red. Y para concluir este tipo de sistema escasea de escalabilidad y de robustez.

### 2.2.2 **Arquitectura Descentralizada**

En la arquitectura descentralizada, los sistemas no hacen uso de mecanismo centralizados, tales como servidores centralizados o “súper nodos” para proveer o coordinar los servicios que la red debe de ofrecer al resto de usuarios de esta. En esta arquitectura todos los participantes tienen los mismos derechos y obligaciones, todos los nodos pueden interactuar como cliente o servidor. A su vez, ninguno tiene la obligación de permanecer en la red, cada uno de ellos puede marcharse de la red en cualquier momento sin afectar en gran medida el funcionamiento del sistema. Al no haber un servidor central, obviamente, el ataque de *single point failure* ya no es posible, debido a que todas los ficheros y servicios que se ofrecen están distribuidos a través de la red, y que ningún nodo

es indispensable. La red tiene una gran resistencia al fallo técnicos de forma parcial de la red y de ataques maliciosos.

Al no haber un nodo central que ayude a la red a localizar los nodos, la forma de comunicación consiste en que el nodo que realiza un *request* les envía un mensaje a sus vecinos para ver si ellos pueden atendérsela.

A raíz de las posibles distribuciones que podían adquirir las redes en esta arquitectura surgen dos variantes de arquitectura descentralizada.

- **Sistema P2P desestructurado.** En este tipo de sistemas el contenido se almacena en cada nodo y no se sigue ningún tipo de relación jerárquica entre nodos. Este tipo de estructura también es conocido como malla.

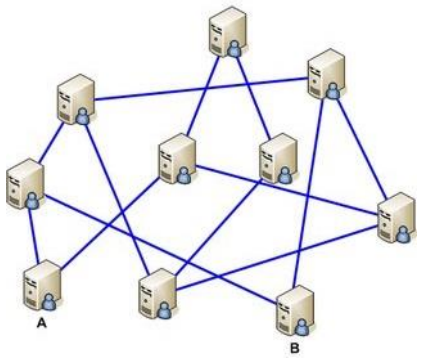


*Figura 5 Red P2P desestructurada*

En este tipo de sistemas, el modo empleado para la comunicación consiste en comprobar en cada nodo qué información tiene y comprobar que es la solicitada. Es por ello por lo que muchos sistemas P2P desestructurados han optado por un sistema de envío de mensajes por inundación de la red (broadcast-based). Gracias a esto se obtiene la ventaja de que es posible comprobar cada nodo de una forma más rápida en un periodo corto de tiempo. Pero esto provoca llenar la red de un gran tráfico de mensajes, debido a esto en este tipo de redes, para evitar la saturación de mensajes, cada uno de ellos lleva un campo *Time-To-Live* (TTL), que es un contador que se va decrementado por cada nodo visitado. Cuando este campo llega al valor 0 el mensaje se descarta de la red y no se transmite más.

Debido a todo esto, el sistema P2P descentralizado es fácil de implementar, pero hay que tener en cuenta que es ineficiente y tiene un alto consumo de ancho de banda.

- **Sistema P2P estructurado.** Debido a todos los problemas que plantea los sistemas P2P desestructurados, surge esta variante, en la que hay un mecanismo para determinar la localización de los ficheros solicitados en la red. Estos archivos son almacenados en localizaciones específicas. Donde los ficheros son mapeados entre los ficheros de la red y el peer que lo almacena. Con esto se consigue que, dada una petición, la localización del fichero puede ser obtenida rápidamente, de forma que no hay que visitar nodos que no estén relacionados y por lo tanto la eficiencia de la petición aumenta de forma importante. Pero la pega de este tipo de red es que los nodos deben mantener la estructura de datos que se haya diseñado para facilitar el enrutamiento. Pero como se ha mencionado previamente cada nodo de la red puede abandonarla o unirse en cualquier momento, y en este tipo de sistema estructurado los nodos deben reagruparse para mantener la jerarquía establecida y la información de enrutamiento se debe de actualizar y el coste de esto es bastante alto.

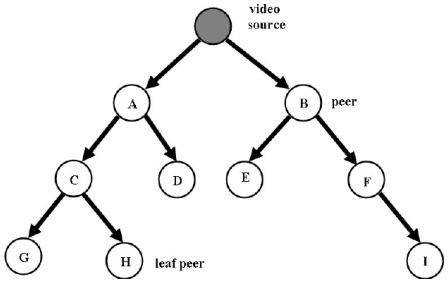


*Figura 6 Red P2P descentralizada jerárquica*

### 2.2.3 **Arquitectura Híbrida**

En cuanto a la arquitectura híbrida, como se ha mencionado antes se basa tanto en la arquitectura centralizada como en la descentralizada, por lo que tiene características de ambas. Las redes híbridas tratan de mantener una escalabilidad similar a la arquitectura descentralizada, no hay servidores centrales en estas redes, sino que, los nodos que tenga un poder computación mayor serán ascendidos a súper nodos. Estos nodos harán las funciones que hacía el servidor central en las redes centralizadas. De esta forma la localización

de recursos puede hacerse de forma descentralizada con técnicas centralizadas (preguntándole al súper nodo) beneficiándose así de las dos arquitecturas.

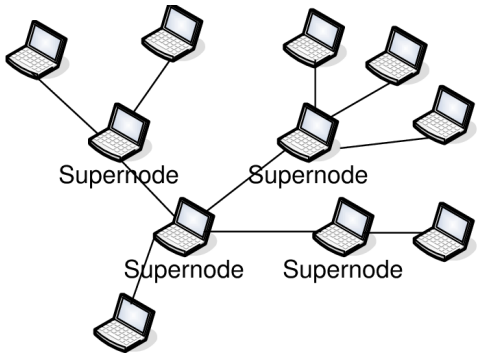


*Figura 7 Red P2P híbrida*

A diferencia de la arquitectura centralizada, en la que si el servidor central es desconectado o sufre un ataque de *single point failure* la red deja de poder funcionar, en la arquitectura híbrida, si un super peer se marcha de la red, esta puede nombrar a otro y que reemplace al nodo desconectado.

## 2.3 **Routing en redes P2P**

Una de las claves de las redes peer-to-peer es el routing de los mensajes o request. Para localizar los recursos deseados, cada nodo debería ser capaz de transmitir las peticiones a un rango de nodos vecinos que estén más cerca al destino que cualquier otro nodo. El diseño de estos protocolos de enrutamiento es uno de los problemas más estudiados e investigados. Hay tres diferentes aproximaciones dependiendo de la arquitectura empleada.

### 2.3.1 **Routing en redes P2P desestructuradas**

Las primeras redes P2P que surgieron eran desestructuradas y los nodos en estos sistemas eran totalmente autónomos. Por lo que no había ninguna topología fija para ellos. En una red P2P desestructurada, cada nodo suele almacenar su propia data y mantiene unas conexiones con otros nodos conocidos como vecinos.

La ventaja en estos tipos de sistemas es que permiten una alta autonomía para los nodos y requiere de un bajo coste de mantenimiento. Pero esta ventaja viene con un alto coste de procesamiento de cola de mensajes, debido a que los nodos no tienen un conocimiento del resto de la red, por lo tanto, hay una inundación de la red por petición.

Debido a esto surgen varias estrategias de enrutamiento para tratar de solucionar este problema.

- **Breadth-First Search**. El Breadth-First Searh o BFS es una técnica sencilla en las redes desestructuradas, donde la profundidad de búsqueda es definida por un parámetro D, que denota el TTL del mensaje a enviar. Cuando un nodo recibe el mensaje este decremente el TTL y procesa la petición. Si no es el nodo buscado por el mensaje, este simplemente hace un *broadcast* del mensaje para todos sus vecinos excepto para aquel que le ha enviado el mensaje. Esto se repite hasta que se alcanza TTL es igual a cero entonces se descarta el mensaje.
- **Depth-First Search**. El Depth-First Search o DFS es una técnica de enrutamiento para redes desestructuradas, donde a la par que BFS también se define le parámetro D, para denotar el TTL del mensaje a enviar. Pero a diferencia del BFS, el DFS selecciona un nodo que considera el más prometedor y le envía la petición a ese nodo. Si al cabo del tiempo el nodo no recibe respuesta, le envía la petición al siguiente nodo que el considera el más prometedor y así sucesivamente hasta recibir respuesta o quedarse sin nodos vecinos. En el caso en el que la petición llegue al nodo con la respuesta para la petición, este envía el mensaje siguiendo la ruta inversa por cada nodo hasta llegar al origen.

### 2.3.2 **Routing en redes P2P estructuradas**

A diferencia de las redes P2P desestructuradas, como se mencionó en la descripción de las redes P2P estructuradas, este tipo de redes los nodos participantes deben de organizarse en una topología definida. Esto quiere decir que cuando un nodo se une al sistema, debe seguir una serie de procesos para establecer su posición en la red de acuerdo con la topología que el sistema adopta. La ventaja de este tipo de redes es que el sistema es capaz de indexar los datos de una forma que cuando se hagan peticiones, la información que se solicite sea más fácil de encontrar. Este tipo de sistemas provee de una búsqueda eficiente y efectiva.

Muchas de las redes P2P estructuradas son capaces de dar respuesta a las peticiones con un coste medio de $O(\log N)$ pasos, donde $N$ es el número de nodos en la red en el que cada paso toma exactamente un mensaje. Esto resultado implica que los problemas de las redes P2P desestructuradas son eliminados totalmente en estas redes P2P estructuradas. La desventaja de este tipo de red es la necesidad de un alto coste de mantenimiento de la topología de esta. Dentro de este tipo de redes se definen categorías principales basado en la descripción de la estructura de esta: sistemas basados en tablas de hash distribuidas y sistemas basados en árbol entre otros.

- Los sistemas basados en tablas de hash distribuidas usan como su nombre indica tablas de hash distribuidas (DHTs) para organizar los nodos y el índice de los datos. En este tipo de sistemas, cada nodo participante es responsable de un rango de valores y cada objeto de los datos es asignado a un valor obtenido mediante una función uniforme de hashing como puede ser SHA-1. Es por ello que este tipo de sistemas son capaces de tramitar peticiones eficientemente. Este tipo de redes heredan buenas propiedades de este tipo de hashing, como por ejemplo la distribución uniforme de la carga de la red en los nodos y así garantizar su buen funcionamiento. La debilidad que tiene este tipo de sistemas es que no son capaces de soportar una cola ordenada de peticiones, debido a que el hashing uniforme elimina el orden de los datos.
- Los sistemas basados en árbol emplean diferentes tipos de árboles para indexar los datos. El objetivo de usar estructuras de árboles es poder soportar colas ordenadas de peticiones de forma eficiente. Esta técnica consiste en que cuando un nodo solicita datos de la red, esta petición se la hace a su nodo padre, el nodo padre comprueba si el tiene esa información, si este la tiene se la devuelve al que la ha solicitado, si no la tiene, pregunta a su padre y así sucesivamente hasta que la información se encuentra y se devuelve al nodo que la ha solicitado y en el que los nodos que han preguntado hacen un *backup* de la información para el futuro por si otro nodo solicita la información. En el caso de llegar al punto de llegar al nodo en el nivel máximo de la red y la información no la tiene indexada tampoco, el paquete se descarta ya que quiere decir que

esa información no está en la red, ya que si el padre de la red entera (nodo en el mayor nivel) no la tiene es que en la red no existe, ya que este nodo suele tener todos los datos de la red indexada.

### 2.3.3 **Routing en redes P2P híbridas**

Las redes P2P desestructuradas inundan la red de peticiones para poder tratarlas, mientras que en las redes estructuradas son capaces de localizar la información solicitada dentro de un rango limitado de saltos por los nodos.

Las redes aprovechan las ventajas de ambos tipos de redes para solventar sus fallos, y así es como los sistemas híbridos fueron inventados.

Los sistemas tradicionales tratan los nodos por igual y no se fija en sus diferencias. Pero al contrario que estas, las redes P2P se fija en sus diferencias para promocionar a aquellos nodos con mayor potencia a súper nodos, y establecer la red de forma jerárquica, donde los súper nodos están en el nivel superior y los nodos estándares están en el nivel inferior. Cada nodo estándar se conecta solo al súper nodo asignado o a los otros nodos que pertenecen al mismo súper nodo.

En este tipo de redes, cada súper nodo solo atiende las peticiones de los nodos asignados a él. El protocolo de comunicación en este tipo de redes es muy sencillo, sigue solo 4 pasos. En el primer paso, un nodo cliente envía la petición a su súper nodo asignado. Segundo, el súper nodo busca en su directorio para determinar que cliente asignado a él o qué súper nodo tiene la información. Tercero, la petición es enviada a otro súper nodo que si puede que uno de sus clientes contenga la información. Por último, se la pasa la dirección IP del nodo que ha solicitado la información al nodo que es capaz de contestar la petición y este se comunica de forma privada con el nodo solicitante y le pasa los datos solicitados.

# 3. Tecnologías relacionadas

En la actualidad y, dada la situación de globalización tecnológica que está experimentando la humanidad, se ha visto magnificado el uso de las redes, tal y como se ha visto en el Capítulo 1. Dada la importancia que estas están teniendo en dicho avance tecnológico, múltiples estudios han surgido con el fin de estudiar el funcionamiento y las bondades de las redes.

Algunos ejemplos de los estudios realizados han surgido para conocer las diferentes modalidades y la eficiencia de estas, como las redes Cliente/Servidor [Svobodova Liba, 1985], [G. S. Hura, 1995] o como las redes Peer to Peer [G. Fox, 2001], [X. Yang, G. de Veciana, 2004] entre otros. Pero, debido a la importancia de la eficiencia de estas, también se han realizado estudios comparativos entre diferentes redes [Robin Jan Maly, 2004].

[R. Runhua and Shun Zhang, 2017] mostraron un esquema de transferencia de clave en redes cliente-servidor, en el que definían dos distintas versiones de autenticación (autenticación basada en identidad y autenticación anónima) en la que el usuario almacenaba su clave privada en secreto. Con esto se obtenía una mejor eficiencia por parte del servidor a la hora de manejar las claves secretas de los usuarios de la red.

Por otro lado, con las redes P2P también se han realizado una serie de aplicaciones.

[M. Ripeanu, 2001] definió el comportamiento de Gnutella. Una aplicación de transmisión de ficheros basados en P2P. Destaca que la aplicación es abierta y descentralizada. En la que cada red Gnutella mantiene conexiones TCP abiertas entre los nodos que conforman la red, en el que los mensajes se propagan mediante una inundación de la red (broadcast). En este estudio Ripeanu concluye que el volumen de tráfico que se genera en la red es el mayor obstáculo para obtener un mejor escalado y un despliegue más amplio.

[Siu Man Lui, Sai Ho Kwok, 2002] precisó la conducta de varias aplicaciones P2P y entre ellas define Napster. Napster era una aplicación que, igual que Gnutella era de transmisión de ficheros basado en P2P, la diferencia respecto de Gnutella es Napster era una red P2P híbrida, los nodos (peers) aún seguían conteniendo la información,

pero esta red tenía un servidor central que contenía una lista con los peers y su dirección, obteniendo así un menor consumo de la red y una mayor escalabilidad, pero convirtiéndose en una red dependiente de un servidor.

[B. Fan, et al. 2006] concretó el funcionamiento de BitTorrent como aplicación P2P, La idea principal es librar a la red de un servidor central, pero con ello se daría una alta demanda de los clientes al nodo que tenga el fichero solicitado, es por ello por lo que BitTorrent divide ese fichero en otros más pequeños y que cada vez que un nodo descarga uno de los fragmentos inmediatamente comienza el también a compartirlo en la red. También se diferencias distintos tipos de peers.

**Tracker** que es "servidor central" que lleva un listado de qué peers y seeds están compartiendo o descargando el fichero.

**Peer** es el usuario que está descargando y compartiendo el fichero solicitado.

**Seed** es el nodo de la red que ya terminó de descargar el fichero y se dedica a compartirlo en la red, para el funcionamiento de BitTorrent es necesario al menos un seed.

[Delgado-Segura, et al. 2018] expusieron las Criptomonedas, tales como Bitcoin, como un nuevo paradigma de las redes P2P, así como el uso de estos tipos de redes para mantener este sistema de las BlockChain lo más descentralizado posible y dada su gran resiliencia y seguridad. El estudio concluyó que este tipo de redes suponen un nuevo paradigma para las redes P2P debido a las propiedades que estas deben proveer: seguridad y fiabilidad.

# 4. OMNET++

OMNeT++ es un entorno de simulación extensible, modular y basado en componentes con librerías de simulación y *frameworks* escritos en C++, en el que estos módulos están pensados en ser reutilizables. A su vez hay extensiones para simulaciones en tiempo real, simulaciones de red y además OMNeT++ soporta otros lenguajes además de C++ soporta Java y C#.

OMNeT++ fue creado con el objetivo en mente de simular redes de comunicación y otros sistemas distribuidos, pero en vez de ser un simulador específico para ciertas simulaciones fue diseñado para ser lo más generalista posible. Desde entonces OMNeT++ ha sido utilizado en numerosos campos de simulación de redes (simulaciones de redes con cola, redes inalámbricas, redes P2P, etc.)

En cuanto a la arquitectura del simulador se puede decir que OMNeT++ es un entorno de simulación genérico que soporta solo capas de la red ordenadas y soporta simulación discreta en el que los módulos se comunican mediante el envío de mensajes. Además, OMNeT++ soporta simulación distribuidas paralelas.

Los módulos simples o componentes diseñados dentro del entorno de simulación se pueden agrupar para crear módulos complejos y así sucesivamente. El nivel de jerarquía es ilimitado. Estos módulos se comunican entre sí enviando mensajes a través de las rutas definidas entrando o saliendo a través de las puertas o *gates* establecidos para cada módulo o directamente si los módulos son *Wireless*.

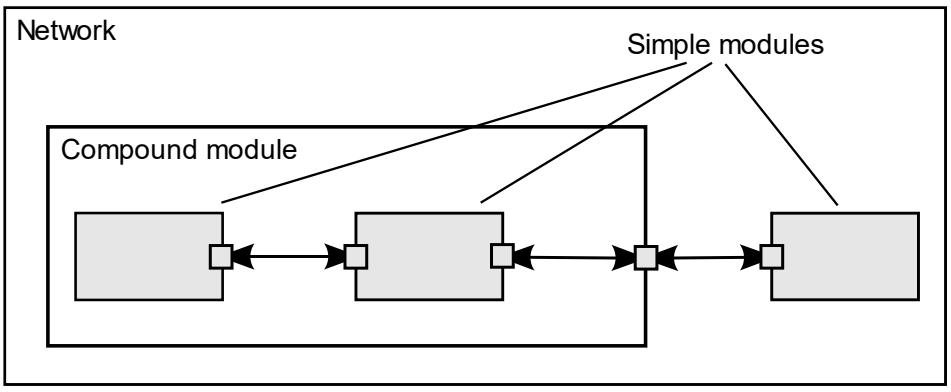


*Figura 8 Modularización OMNeT++*

Además, las conexiones creadas entre los módulos para comunicarse pueden ser configuradas para parametrizar la velocidad del enlace, el retardo, la tasa de error de envío etc. Todo esto con el fin de crear una simulación más realista.

Es posible también parametrizar campos de los propios módulos para modificar el comportamiento de estos. Los parámetros pueden definir, por ejemplo, dentro de un módulo compuesto el número de *gates* que tiene cuantos submódulos tiene, etc. Todo es posible de controlar a través de los ficheros .ned.

Los ficheros .ned son los archivos de descripción de la red (Network Description), en este archivo se almacena la descripción de la estructura de la red, este fichero es usado para describir la estructura lógica de la red que será simulada más adelante. Los archivos NED tienen una estructura jerárquica, esto permite crear estructuras complejas en otro fichero NED y poder reutilizarlo en otro.

```
network Tictoc11
{
    @display("bgb=345,356");
    types:
        channel Channel extends ned.DelayChannel{
            delay = 100ms;
        }
    submodules:
        tic[6]: Txc11 {
            @display("p=156,173");
        }
    connections:
        tic[0].out++ --> Channel --> tic[1].in++;
        tic[0].in++ <-- Channel <-- tic[1].out++;

        tic[1].out++ --> Channel --> tic[2].in++;
        tic[1].in++ <-- Channel <-- tic[2].out++;

        tic[1].out++ --> Channel --> tic[4].in++;
        tic[1].in++ <-- Channel <-- tic[4].out++;

        tic[3].out++ --> Channel --> tic[4].in++;
        tic[3].in++ <-- Channel <-- tic[4].out++;

        tic[4].out++ --> Channel --> tic[5].in++;
        tic[4].in++ <-- Channel <-- tic[5].out++;
}
```

*Figura 9 Fichero .ned OMNeT++*

Los datos necesarios para la configuración de la red se encuentran en un fichero de configuración .ini por lo general este fichero suele denominarse omnet.ini. Este fichero contiene las opciones que controlan como la simulación se va a ejecutar y contiene los parámetros a tener en cuenta durante la simulación. Además, también es posible dar valor a los parámetros establecidos de los módulos descritos en los ficheros NED. En este fichero se pueden definir diferentes escenarios de simulación, para ello habrá que definir distintas secciones estás secciones van por lo general entre corchetes ([]). Todo fichero .ini debe tener una sección general que se define como [General] donde se especifican los parámetros generales que afecten a todas

las redes, también es posible crear secciones específicas para casos concretos de estudios la cabecera de estás secciones se definen como [Config "nombre de la sección"]. Además, se puede hacer que si configuraciones de una sección se ejecuten en otra se puede hacer que una sección herede de otra a través de un extends = "nombre de la sección a heredar".

OMNeT++ permite realizar iteraciones para poder ejecutar múltiples veces una simulación.

```
[Config HybridNetwork]
network = HybridNetwork
HybridNetwork.numNodes = ${10, 20, 30, 50, 100, 200, 500, 1000}
HybridNetwork.numPackages = 20
```

*Figura 10 Configuración fichero .ini con iteraciones*

Como se puede apreciar en la figura la simulación se ejecutar varias veces aumentando el número de nodos de la red por cada iteración.

A su vez como se ha mencionado antes OMNeT++ permite realizar iteraciones paralelas.

```
HybridNetwork.numNodes = ${10, 20, 30, 50, 100, 200, 500, 1000}
HybridNetwork.numPackages = ${10, 20, 30, 50, 100, 200, 300, 350}
```

*Figura 11 Definición Iteración paralela*

Tal y como se observa en la figura 11, se ha definido un bucle iterativo en el que a diferencia de la figura 10 que realizaba varias simulaciones iterativamente aumentando el número de nodos, en este se realizarán varias simulaciones iterativamente en el que además de ir aumentando el número de nodos que formarán la red, también se aumentará a la par el número de paquetes que se enviaran por la red. Generando así una iteración paralela, que comenzará con 10 nodo y se enviarán 10 paquetes en la red y finalizará con 1000 nodos y se enviarán 350 paquetes.

Finalmente, para poder visualizar los datos, el IDE de OMNeT++ ayuda al usuario a poder realizar esto, lo que hace el simulador es generar una serie de ficheros en el directorio *results* del proyecto en el que genera un fichero .vec y un fichero .sca que son los ficheros que contienen los resultados en la forma vectorial y escalar.

OMNeT lo que hace es que al visualizar estos datos permite “plotearlos” para representar estos resultados más fácilmente.

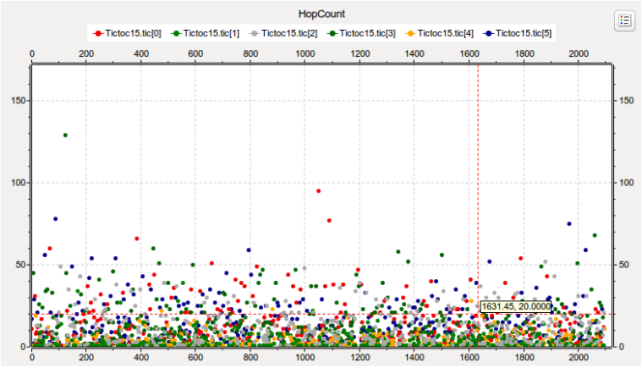


*Figura 12 Visualización de datos en OMNeT++*

# 5. Caso de estudio

En este capítulo se explicará la implementación de las diferentes redes en OMNeT++, el entorno de simulación introducido en el capítulo anterior para simular la lógica de funcionamiento de las tres topologías a estudiar en este proyecto.

Cabe destacar que, en esta simulación de las distintas redes, no se llegará a profundizar en el funcionamiento de estas. Se realizará un estudio como se ha mencionado antes de la lógica de funcionamiento en cuanto al envío de paquetes.

Más adelante, en la sección 5.1, se explicará la implementación de las distintas redes en el simulador y su funcionamiento mediante código.

Después, en la sección 5.2, se definirán los casos de prueba a los que se someterán las redes, estableciendo número de nodos, velocidad de los canales, etc. Y se explicarán los parámetros que se establecerán en la configuración de la red.

## 5.1 **Establecimiento de las redes P2P**

### 5.1.1 **Topología malla**

En este apartado se procederá a explicar la implementación de la red P2P siguiendo la arquitectura de malla.

Como se describió en el capítulo 2, una red P2P de malla, es un tipo de red P2P descentralizada no jerárquica plana, es decir, todos los nodos están al mismo nivel, teniendo la misma responsabilidad.

Dentro del entorno OMNeT++ se procederá a describir la red dentro de un fichero .ned.

Los nodos que formarán parte de la red contarán con una lista de *gates* de tipo *inout* para establecer los canales de comunicación el resto de los nodos.

```
gates:
    inout g[];
```

*Figura 13 Definición gates en topología malla*

En el caso de la definición de la red se definirán dos parámetros esenciales para establecer los canales de comunicación, que son el número de nodos que formarán la red, que por defecto tiene el valor de 100 y un valor entero que será el número de vecinos que tendrá un nodo, que por defecto tiene el valor de 2.

```
int numNodes @prompt("Number of node") = default(100);
int maxconnections = default(2);
```

*Figura 14 Parámetros para la definición de la red con topología malla*

Para establecer las conexiones entre los nodos, por cada nodo se establece una conexión con los siguientes nodos después de él, hasta llegar al máximo de vecinos pasibles para ese nodo.

```
submodules:
    node[numNodes]: MeshNode;
connections allowunconnected:
    for i=0..(numNodes-1), for j=(i+1)..(i+maxconnections) {
        node[i].g++ <--> NetworkConnection <--> node[j].g++ if j <= (numNodes-1);
    }
```

*Figura 15 Establecimiento conexión entre nodos topología malla*

Es por ello por lo que, a mayor valor asignado al número de vecinos posibles, más rápida será la red, ya que, al hacer difusión a los vecinos, se hará a todos los nodos de la red, por lo que llegará al nodo requerido, mientras que, a menor valor de conexiones máximas a vecinos, mayor cantidad de recorrido se deberá hacer, siendo esto el caso más realista, ya que de normal no todos los nodos se conocen entre sí, si no que tienen que ir preguntando a sus vecinos. A consecuencia de esto, a menor número de vecinos posibles para cada nodo más lenta será el tratamiento de las peticiones en esta red y por lo tanto menos escalable.

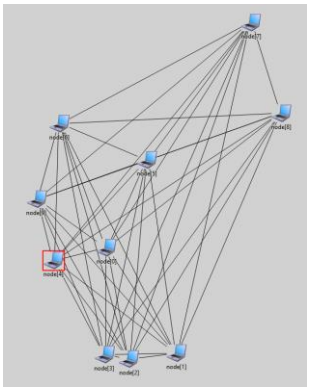

*Figura 16 Red P2P malla perfecta*

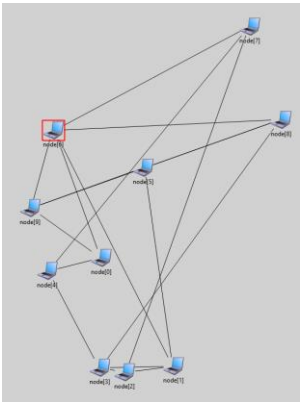

*Figura 17 Red P2P malla estándar*

Todo esto es en cuanto a la definición de la red, pero sin implementar ninguna lógica de funcionamiento. El comportamiento que debería seguir esta red se ha definido en un fichero .c, en el que se describe el comportamiento de cada nodo.

El primer paso que hacen los nodos al iniciarse la red es generar un paquete propio simulando que fuera el fragmento de datos que tiene el de la red.

```
packet = generatePacket();
```

*Figura 18 generación de paquete*

Después se elige aleatoriamente un nodo que solicite un paquete al azar, generando el mensaje a transmitir por la red estableciendo el paquete deseado, el origen, y el Time To Live del paquete. Y programa su salida con un retardo programado.

```
if(k == getIndex()){
    EV<< getIndex();
    MessageNode *msg = generateMessage();
    EV <<msg ->getPacketReq();
    scheduleAt(0.0 + j, msg);
    PackageSend++;
}
```

*Figura 19 Envío de petición a la red*

Cabe destacar que la función *scheduleAt* envía un mensaje al propio nodo para iniciar la comunicación por la red.

En cuanto al algoritmo de comunicación empleado, cabe recordar que para las redes P2P desestructuradas se explicaron dos algoritmos de comunicación que se podían emplear el BFS y el DFS. Para la implementación de esta red se ha optado por hacer uso del BFS, por lo que, por cada mensaje recibido, primero se

comprobará su TTL, si es mayor que 0 el mensaje se seguirá transmitiendo, si no, es descartado.

```
if(msg->getTTL() == 0){
    delete msg;
    EV << "mensaje descartado TTL caducado";
    return;
}
```

*Figura 20 Descarte del paquete al agotarse su TTL*

Para el mensaje, su TTL es mayor que 0, se comprueba que el nodo que ha recibido el mensaje comprueba si él tiene el paquete solicitado, si no lo tiene hace *broadcast* del mensaje recibido enviándolo a todos sus vecinos excepto al que se lo ha enviado a él, tal y como el algoritmo BFS describe.

```
int n = gateSize("g");

arrival = msg->getArrivalGate()->getIndex();

for(int j=0; j< n; j++){
    if(j != arrival){
        send(msg->dup(), "g$o", j);
    }
}
```

*Figura 21 Broadcast a los vecinos*

Si el mensaje llega a destino, este envía devuelta el paquete solicitado, siguiendo el mismo camino que ha tomado el mensaje hasta llegar al destino, es decir, pasa por los mismos nodos que ha pasado el mensaje hasta llegar al nodo con el paquete solicitado.

### 5.1.2 **Topología árbol**

En este apartado se procederá a explicar la implementación de la red P2P siguiendo la arquitectura de árbol.

Como previamente se explicó en el capítulo 2, una red P2P de tipo árbol, es una red P2P descentralizada, por lo que la información está repartida por la red, y a su vez es jerárquica, eso quiere decir que no todos los están al mismo nivel.

Como se mencionó en la descripción de la red P2P de tipo malla, la descripción de esta en el entorno de simulación se realizará en un fichero .ned.

A diferencia de la red previamente descrita, aquí los nodos no tienen una lista de *gates*, sino que se tienen tan solo 3, una para establecer comunicación con el nodo padre y otras dos para establecer con sus hijos en caso de que los tenga.

```
simple TreeNode extends Node{

    gates:
        inout toParent;
        inout leftChild;
        inout rightChild;
}
```

*Figura 22 Definición de gates de los nodos del árbol*

En cuanto a la definición de la red, se establecerán dos parámetros esenciales para el funcionamiento de la red, el primero que es común a todas las redes, es el número de nodos que formarán la red, y el segundo es la altura del árbol que se creará.

```
parameters:
    int height @prompt("Height of the tree") = default(5);
    int numNodes = default(2^height-1);
```

*Figura 23 Parámetros para formar la red*

En cuanto a la formación de la red, en esta red también se tendrá una lista con los nodos que la formen, siendo el nodo[0] el nodo raíz del árbol del que se conectarán el nodo[2*0+1] y nodo[2*1+2] como hijos, para el nodo 1 se conectarán como hijos los nodos 2*1+1 y 2*1+2 y así sucesivamente para cada nodo i se conectarán los nodo 2*i+1 y 2*i+2 como sus hijos.

```
submodules:
    node[numNodes]: TreeNode;
connections allowunconnected:
    for i=0..2^(height-1)-2 {
        node[i].leftChild <--> NetworkConnection <-->node[2*i+1].toParent;
        node[i].rightChild <--> NetworkConnection <--> node[2*i+2].toParent;
    }
```

*Figura 24 Establecimiento de la red P2P con Topología árbol*

Como se puede apreciar en la figura de abajo se muestra una red P2P siguiendo está arquitectura definida en el simulador, estableciendo un árbol con una altura de 5 y 31 nodos.

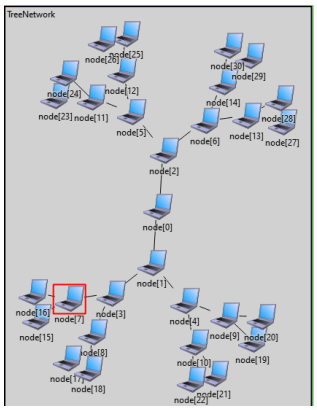


*Figura 25 Red en árbol ya estructurada*

Con todo esto la red ya estaría definida. En cuanto al comportamiento que los nodos deben seguir, está definido en otro fichero escrito en C en el que como se explicó con la red anterior, se define el comportamiento de los nodos a la hora de enviar paquetes o mensajes.

El primer paso que hacen los nodos es que si eres el nodo raíz (nodo en la posición de 0 de la lista de nodos) genera el paquete que será solicitado por el resto de los nodos.

```
if(getIndex() == 0){
    packet = generatePacket();
    packets[i] = packet;
}
```

*Figura 26 Generación de la información en la raíz*

Después se elige aleatoriamente un nodo que solicite un paquete al azar, generando el mensaje a transmitir por la red estableciendo el paquete deseado, el origen. Y programa su salida con un retardo programado.

```
if(k == getIndex()){
    EV<< getIndex();
    MessageNode *msg = generateMessage();
    EV <<msg ->getPacketReq();
    scheduleAt(0.0 + j, msg);
    PackageSend++;
}
```

*Figura 27 Envío de la petición al padre*

Una vez se inicia el protocolo de comunicación, el nodo origen de la solicitud, le envía el mensaje a su nodo padre a través del canal establecido. Una vez el padre los recibe, lo primero que hace es comprobar si él tiene el paquete solicitado.

```
if(!checkIfLocal(msg ->getPacketReq())){
```

*Figura 28 Comprobación de si se tiene la información deseada*

En caso de que lo tenga le envía el paquete solicitado al nodo que lo había solicitado, en caso negativo este nodo enviará el mensaje a su padre.

```
if(gate("toParent$o")->isConnected()){
    send(msg, "toParent$o");
```

*Figura 29 Transmisión de la solicitud al nodo padre*

En el caso de llegar al nodo raíz y que este no tenga el paquete solicitado, querrá decir que ese paquete no está en la red. Y por lo tanto descartará el mensaje.

```
if(gate("toParent$o")->isConnected()){
    send(msg, "toParent$o");
}else{
    delete msg;
    EV << "required Packet not Exist";
}
```

*Figura 30 Descarte del paquete por el nodo ráiz*

Pero en caso de que sí exista el paquete, el paquete retorna por la misma ruta de nodos por la que vino y, según vaya pasando por los nodos, estos se harán una copia del paquete para que, en caso de que alguno de sus nodos hijos los solicite, puedan ellos mismos enviar el paquete sin tener que llegar hasta el nodo raíz de la red.

```
if(!checkIfLocal(pckt.getPName())){
    addPacketToColection(pckt);
}
```

*Figura 31 Añadir información en local*

### 5.1.3 **Topología híbrido**

En este apartado se procederá a explicar la implementación de la red P2P siguiendo la arquitectura híbrida.

Como se definió en el capítulo 2, una red P2P híbrida, es un tipo de red que aprovecha las ventajas tanto de las centralizadas como de las descentralizadas. La información está distribuida por la red y hay un súper nodo que sabe que nodo contiene que información.

Para simular el funcionamiento de esta red en OMNeT++, primero se procederá a describirla en un fichero .ned.

En esta red, a los nodos se les establecerá una *gate*, que es con la que se establecerá el canal de comunicación con el súper nodo.

```
gates:
    inout toSuperNode;
```

*Figura 32 Definición de gates para los nodos estándar*

Mientras que el súper nodo tiene una lista de *gates* en las que se irán añadiendo aquellas que establezcan comunicación con los nodos.

```
gates:
    inout g[];
```

*Figura 33 Definición de gates para los nodos súper nodos*

Una vez definidos tanto los nodos básicos como el súper nodo, se procederá a establecer los canales de conexión de cada nodo con el súper nodo.

```
connections allowunconnected:
    for i=0..(numNodes-1) {
        superNode.g++ <--> NetworkConnection <--> node[i].toSuperNode;
    }
```

*Figura 34 Definición de las conexiones en la red*

Y con esto la red ya estaría definida. Una vez descrita esta, se obtendrá una red como la que se puede apreciar en la figura. En la que se puede apreciar como todos los nodos en un principio tienen conexión con el súper nodo solo.

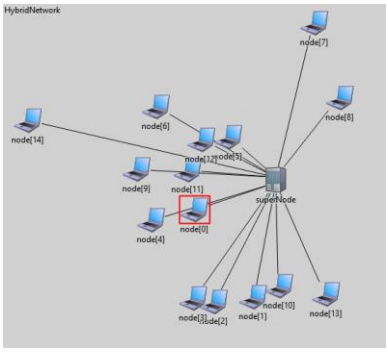


*Figura 35 Ejemplo red P2P híbrida con un súper nodo*

En cuanto a la lógica de funcionamiento a simular, el comportamiento establecido será definido en un fichero escrito en C. Para comenzar, al ser una red con la información distribuida por los nodos, cada uno de estos generará un paquete que simulará la parte de la información de la red que el contiene. Tal y como ocurría en el caso de la red P2P con arquitectura en malla.

```
if(std::strcmp(getFullName(), "superNode") != 0){
    packet = generatePacket();
```

*Figura 36 Generación de la información descentralizada*

Una vez generado los paquetes, cada nodo informa al súper nodo qué paquete contiene, y el súper nodo procede a listar por cual compuerta ha recibido la información de qué fragmento de la información almacena cada nodo.

```
if(ttmsg->getInfo() && std::strcmp(getFullName(), "superNode") == 0){
    NodePackets[ttmsg->getSrc()] = ttmsg->getPacketOwn();
    return;
}
```

*Figura 37 Súper nodo almacena los datos que le envían cada nodo estándar*

Si la simulación es ejecutada se mostrará una lista que a primera vista no parece tener mucho sentido, ya que no se muestra mucha información. Pero si se expande uno de los mensajes almacenados, se puede observar que el súper nodo almacena el nombre del paquete del que un nodo es propietario y, si un nodo solicita ese paquete, también contiene por qué compuerta debería transmitir el mensaje recibido para que el nodo propietario lo recibiera y estableciera comunicación con el nodo solicitante.

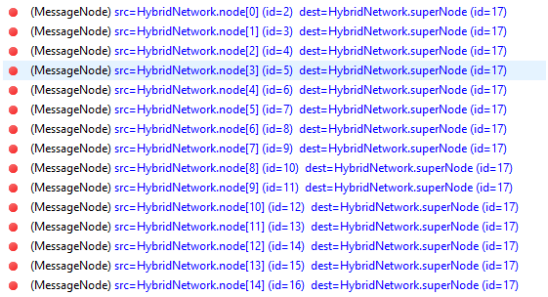


*Figura 38 Lista de la información almacenada por el súper nodo*

```
packetOwn = 'Packet_0' [...] (string)
gateIdToSent = 0 [...] (int)
```

*Figura 39 Información almacenada detallada*

Una vez que el servidor ha *routeado* a todos los nodos y ya sabe en qué nodos está la información distribuida, la red ya está preparada para transmitir. Por lo que, si un nodo solicita información, le enviará un mensaje al súper nodo, solicitándole el paquete deseado. Una vez el súper nodo recibe ese mensaje, este se lo transmite al nodo propietario de dicho paquete solicitado.

```
if(!ttmsg->getInfo() && std::strcmp(getFullName(), "superNode") == 0){
    send(ttmsg, "g$o", ttmsg->getDest());
    PackageSend++;
    return;
}
```

*Figura 40 transmisión de información solicitada desde el súper nodo al nodo con la información requerida*

Una vez el nodo destinatario recibe el paquete, este adjunta al mensaje el paquete solicitado.

```
ttmsg->setMPacket(packet);
```

*Figura 41 Adjuntar la información solicitada al mensaje*

Pero como se definió teóricamente en el capítulo 2 en las redes híbridas, la comunicación se hace directa entre nodos, al hacer un traspaso de información requerida. Para poder establecer comunicación con el nodo solicitante, se establecerá un canal y se crearán compuertas por las que se pueda establecer comunicación todo esto de forma dinámica.

```
std::string channelName = "directo";

cDatarateChannel *channel = cDatarateChannel::create(channelName.c_str());
```

```
std::string emisorGate = randomName(ttmsg->getSrc() + getId());
std::string receptorGate = randomName(ttmsg->getSrc() + getId());
```

*Figura 42 Creación de canal privado entre nodos*

Una vez creados tanto el canal como las compuertas, se debe establecer la conexión.

```
addGate(emisorGate.c_str(), cGate::OUTPUT)->connectTo(getSimulation()->getModule(ttmsg->getSrc())->addGate(receptorGate.c_str(), cGate::INPUT), channel);
```

*Figura 43 Establecimiento de conexión entre los nodos*

Ya establecida la conexión, el nodo propietario del paquete envía el mensaje con la información solicitada. Una vez la información llega al nodo solicitante, este la almacena en su memoria interna. Se procede a borrar los canales y las compuertas creadas para no tenerlas en desuso.

Para borrar las compuertas previamente se deben desconectar.

```
gate(emisorGate.c_str())->disconnect();

gate(receptorGate)->disconnect();
```

*Figura 44 Desconexión de gates*

Y una vez desconectadas ya pueden ser eliminadas.

## 5.2 Configuración de escenarios

Una vez definidas las tres redes a ser sometidas a pruebas se va a definir los distintos casos de estudio. Las principales variables que se van a tener en cuenta son: el número de nodos que formen la red, el ancho de banda de los canales establecidos entre ellos, el número de mensajes promedio que se envían para tratar una petición, y el tiempo promedio en resolver una petición con el fin de así entender que red P2P con diferentes arquitecturas en la más eficiente.

En cuanto al número de nodos que forman la red, hay que tener en cuenta la principal limitación que implica usar en el entorno de simulación OMNeT++, y es el número de nodos. Comparado con otros simuladores, OMNeT++ no permite crear redes de gran magnitud, el simulador permite crear una red con un máximo de 1000 nodos.

| Simulator | Language | Architecture | | Scalability |
|---|---|---|---|---|
| | | P2P structure | Simulator mode | |
| PeerfactSim. KOM | Java | Structured and unstructured overlays | Discrete event | Very high $10^6$ peers for simple overlays $10^5$ peers for complex overlays |
| D-P2P-Sim | Java | - | Discrete event | Extremely high, >400,000 peers |
| ProtoPeer | Java | Structured and unstructured overlays | Discrete event | Approximately 50,000 peers |
| PeerSim | Java | Structured and unstructured overlays | Query cycle or discrete event | Very high $10^6$ nodes using cycle-based, $2.5 * 10^5$ nodes using event-based |
| RealPeer | Java | Structured and unstructured overlays | Discrete event | 20,000 nodes |
| OMNeT++ | C++ | Structured overlays | Discrete event | Low 1000 nodes |
| OverSim | C++ | Structured and unstructured overlays | Discrete event | Medium 100,000 nodes |
| Overlay weaver | Java | Structured overlays | Discrete event | Low 4000 nodes |
| PlanetSim | Java | Structured and unstructured overlays | Discrete event | Medium 100,000 nodes |
| Dnet | C/C++ | – | Discrete event | 10,000 nodes |
| 3LS | Java | Unstructured overlays | Clock-based simulation engine (cycle) | Low 1000 nodes |
| Optimal-sim | Java | – | Discrete event | 10,000 nodes |
| NS-2 | C++ and Object-oriented version of TCL | Structured and unstructured overlays | Discrete event | Low 5000 nodes |

*Figura 45 Comparativa entre simuladores*

Una vez introducido las variables a tratar y la limitación que nos establece OMNeT++, Es posible definir los diferentes escenarios a los que las redes serán sometidas.

En cuanto al número de nodos que formarán la red, se han definido los siguientes escenarios:

1- 25 nodos
2- 50 nodos
3- 100 nodos
4- 200 nodos

En cuanto a la velocidad de la red a la que serán sometidos los diferentes canales de comunicación establecidos, se han definido los siguientes escenarios:

1- 50 kb/s
2- 100 kb/s
3- 1 mb/s
4- 20 mb/s

Todos estos escenarios serán sometidos siempre simulando que en la red se va a tramitar mensajes de 64 KB. Una vez se tiene claramente definidos los distintos escenarios a los que se van a someter las distintas redes descritas, se procederá a describir los distintos escenarios en el fichero .ini dentro de OMNet++.

```
[Config Simulacion25NodosConVelocidad50kbs]
network = HybridNetwork
HybridNetwork.numNodes = 25
**.channel.delay = 10.24s

[Config Simulacion50NodosConVelocidad50kbs]
network = HybridNetwork
HybridNetwork.numNodes = 50
**.channel.delay = 10.24s

[Config Simulacion100NodosConVelocidad50kbs]
network = HybridNetwork
HybridNetwork.numNodes = 100
**.channel.delay = 10.24s
```

*Figura 46 escenarios planteados para la simulación de las redes*

Como se puede apreciar, en ningún momento se está estableciendo la velocidad de transmisión, no aparece ningún campo con ese nombre, eso es debido a lo que se está estableciendo es el delay del canal, es decir, el tiempo que tardará un nodo en enviar el paquete a otro nodo. Ya que el tiempo de transmisión es:

$$t(tx) = \frac{TamañoPaquete}{v(tx)}$$

En el que el tamaño de paquete son los 64KB definidos antes y la velocidad en el ejemplo presentado previamente es de 50Kb/s eso da un valor de 10.24s, que corresponde al tiempo de transmisión que es el valor que se puede encontrar en el ejemplo presentado.

# 6. Análisis de resultados

Una vez definido los distintos escenarios dentro del entorno de simulación y ejecutadas las distintas simulaciones descritas, es hora de ejecutar las simulaciones y realizar un análisis de los distintos resultados obtenidos.

## 6.1 Resultados Topología malla

Comenzando por la primera red definida, siendo esta la red con arquitectura descentralizada siguiendo la topología de malla, si se observa la figura 47.

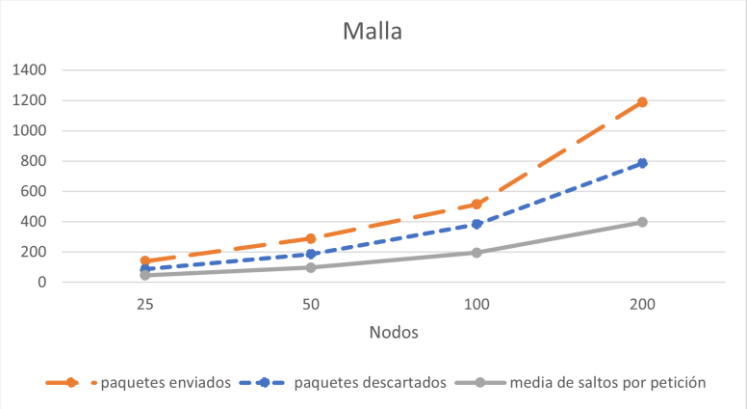


*Figura 47 Información general de los resultados obtenidos en la red con topología en malla*

vemos como, según aumenta el número de nodos de que forman parte de la red, se puede ver cómo tanto el número de nodos, como los paquetes descartados y la media de saltos que realiza un paquete para cumplir la petición del nodo, aumenta según aumentan los nodos.

En cuanto a los paquetes enviados en la red por petición, se observa como aumentan a la par al número de nodos. Esto es debido al algoritmo de búsqueda implementado por este tipo de red, en este caso el BFS, en el que, como ya se ha explicado, al recibir un mensaje, si el par no tiene el paquete solicitado, realiza un *broadcast* del mensaje por sus vecinos. Lo mismo ocurre con los paquetes descartados, que según aumentan los nodos participantes, más paquetes son descartados al no llegar a su objetivo antes de que su TTL llegue a 0. Y con la media de saltos por petición ocurre exactamente lo mismo, al ser la red cada vez más grande, los paquetes requieren pasar por más nodos hasta cumplir su objetivo.

Todo esto se puede corroborar observando el tiempo de simulación obtenido, representando los escenarios distintos. Es por ello por lo que, si se observa la figura 48, se puede ver que la gráfica sigue la misma tendencia que la anterior presentada.

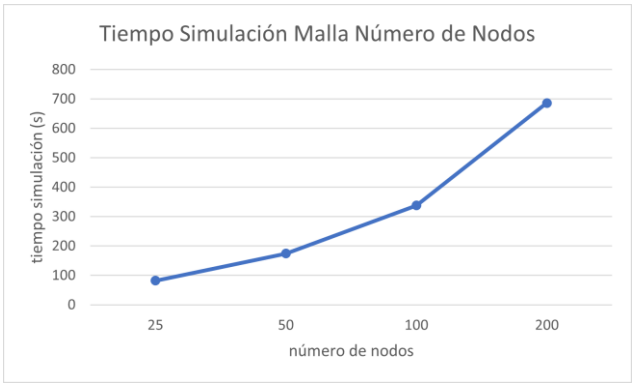


*Figura 48 Tiempo de Simulación según aumenta los nodos en topología en malla*

Una forma de mitigar esta ineficiencia sería si fuera posible aumentar la velocidad de transmisión de los nodos, ya que, tal y como se puede observar en la figura 49 para una red con un número fijo de nodos, si se aumenta la velocidad se puede ver que el tiempo de transmisión disminuye.

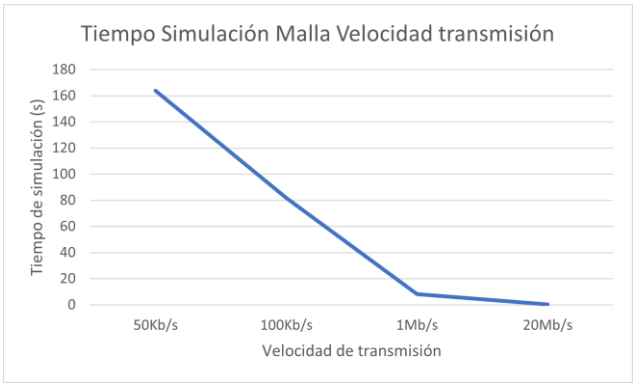


*Figura 49 Tiempo simulación según se aumenta la velocidad de transmisión en topología en malla*

## 6.2 Resultados Topología Árbol

Continuando con la red P2P siguiendo la arquitectura jerárquica de árbol, si se contempla la siguiente figura, se podrá observar que comparada con la anterior red, el número de paquetes enviados es sumamente inferior.

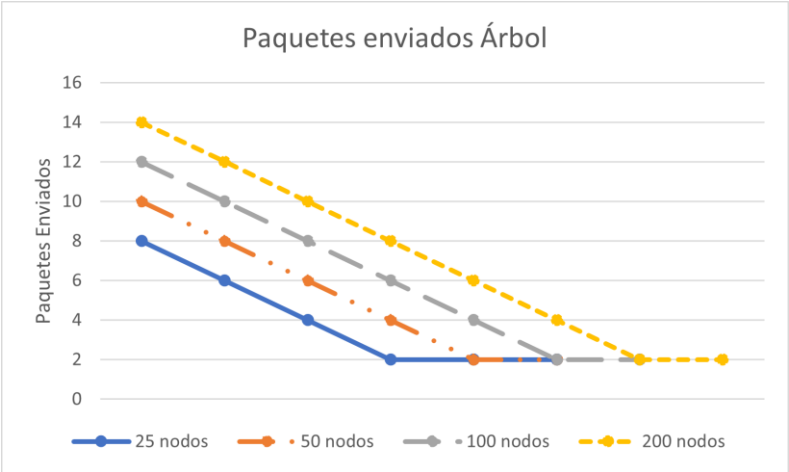


*Figura 50 Información general de los resultados obtenidos en la red con topología en árbol*

Esto es debido, a que en este tipo de redes la información no está distribuida entre todos los nodos, sino que la información al inicio de la red la tiene el nodo raíz de esta, y según se va solicitando, se va distribuyendo por el resto de nodos padres que haya en la red. Es por ello por lo que en un principio, según el nivel de profundidad que tenga la red árbol el nodo tendrá que subir más niveles o no hasta llegar al nodo raíz quien le dará la información, pero según se vaya distribuyendo la información, los nodos solicitantes no tendrán que subir tantos niveles jerárquicos para solicitar la información, hasta el punto en el que todos los nodos padres ya tienen la información y la red se estabilice, y tan solo hagan falta dos mensajes para obtener la información solicitada. El primer mensaje para solicitar la información y el segundo la respuesta del padre con la información.

Tal y como se ha explicado antes, según más nodos formen la red, mayor profundidad tendrá el árbol. Por lo que, mayores niveles deberá subir la petición del nodo en el inicio de la red para obtener la información y por ende mayor tiempo tomará, tal y como muestra la figura 51.

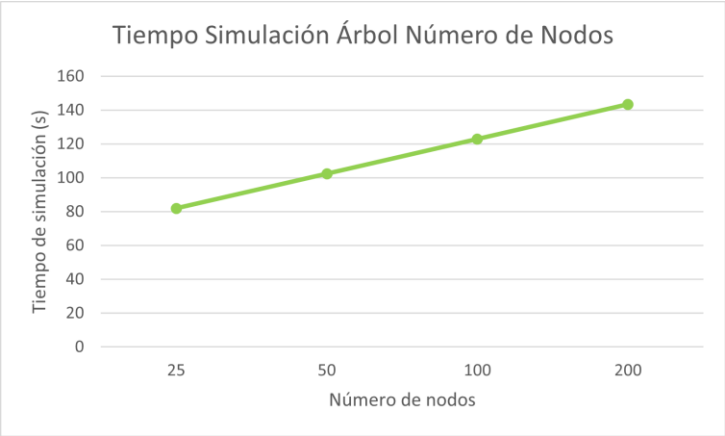


*Figura 51 Tiempo de Simulación según aumenta los nodos en topología en árbol*

Y tal y como se planteó previamente con la red que sigue la arquitectura de malla, la forma de reducir este tiempo es aumentar la velocidad de transmisión de los nodos, con el fin de hacer más eficiente la red. Ya que tal y como se aprecia en la siguiente figura, para una red de este tipo, con un número de nodos fijos, según aumenta la velocidad de transmisión, menor tiempo le toma responder a la solicitud.

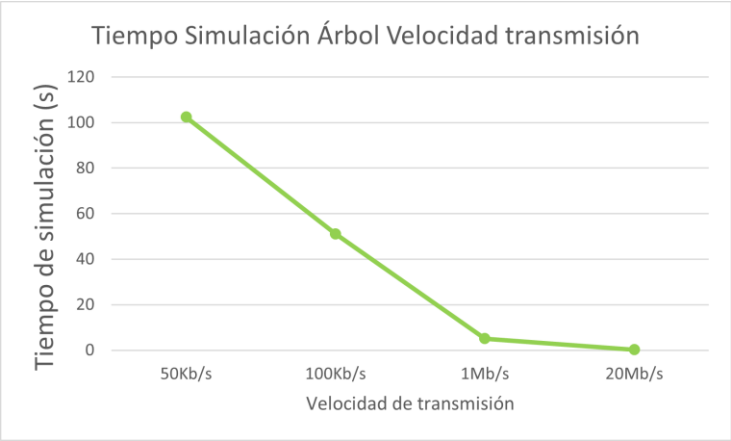


*Figura 52 Tiempo simulación según se aumenta la velocidad de transmisión en topología en árbol*

## 6.3 Resultados Topología Híbrida

La arquitectura híbrida a la par que la arquitectura en árbol comienza con un pico de paquetes enviados y a partir de ahí se estabiliza, como se observa en la figura 53.

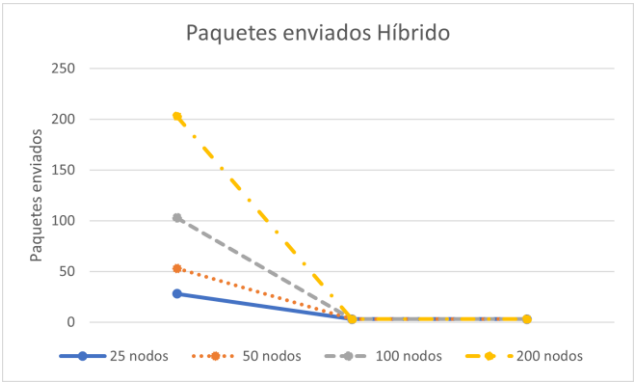


*Figura 53 Información general de los resultados obtenidos en la red con topología híbrida*

El pico de mensajes es mayor comparado con la red con arquitectura en árbol, este pico es debido a que, durante el inicio de la red, los nodos que forman la red deben de comunicarle al súper nodo la información distribuida de la red que contienen, todo esto para facilitar el *routing* al súper nodo. Una vez realizado esto se trata la petición realizada por alguno de los nodos.

Una vez el súper nodo ya tiene ruteados al resto de nodos, la petición tan solo consta de 3 mensajes a mandar. El primero consiste en la petición del nodo al súper nodo, segundo, este envía la solicitud al nodo ruteado con la información y tercero el nodo propietario le envía la información al nodo solicitante.

Si comparamos el tiempo que tarda desde que se inicia una red híbrida hasta que se termina de tratar una petición de un nodo solicitante, se obtiene un curioso resultado. Si se desprecia la demora del servidor en rutear al nodo que le manda la información, se obtiene una gráfica como la siguiente.

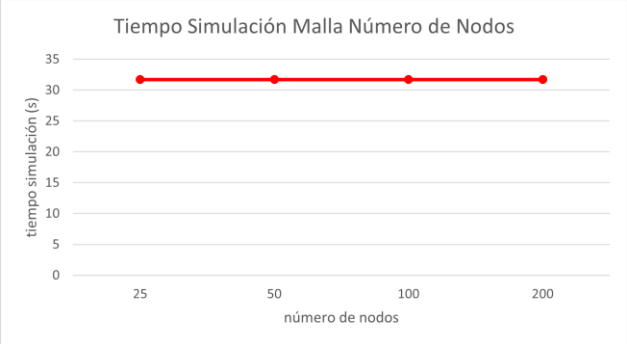


*Figura 54 Tiempo de Simulación según aumenta los nodos en topología híbrida*

El tiempo de simulación, obtenido según se aumenta el número de nodos es igual o prácticamente igual, esto es debido a que, aunque el número de nodos

aumente, el envío de información de cada nodo se hace de forma paralela al resto de envío de mensajes, y el tiempo es el mismo al de un solo mensaje. Este es uno de los motivos de los que este tipo de redes son escalables.

A su vez si se consigue optimizar los canales de transmisión entre los nodos y aumentar la velocidad de transmisión entre nodos, se podría rebajar el tiempo de transmisión bastante, tal y como muestra la figura 55.

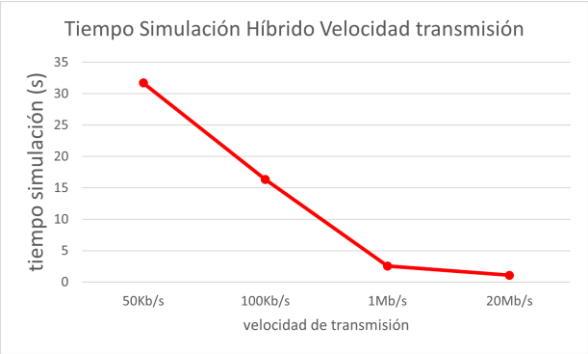


*Figura 55 Tiempo simulación según se aumenta la velocidad de transmisión en topología híbrida*

## 6.4 Comparativa de resultados

Para finalizar si se comparan las tres redes definidas y puestas a simulación, es posible observar, como muestra la siguiente figura, que la primera red es la más lenta de las tres, según se va aumentando el número de nodos de la red y por lo tanto será la menos escalable.

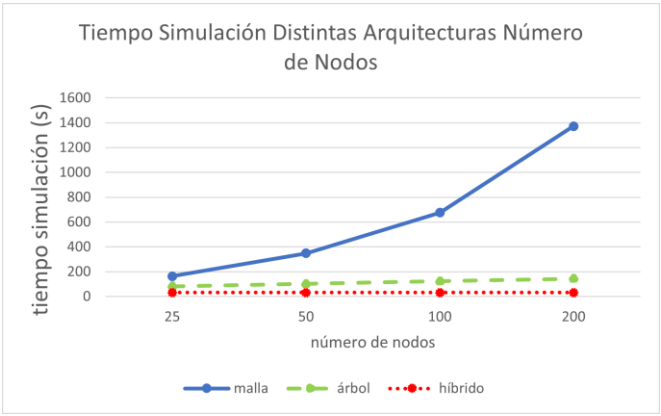


*Figura 56 Comparativa de la tres Topologías del Tiempo simulación según se aumenta el número de nodos*

Y, por lo tanto, debido a que es la más lenta, es la que mayor beneficio obtendría de mejorar los canales de comunicación y la que mayor reducción de tiempo de simulación obtendría.

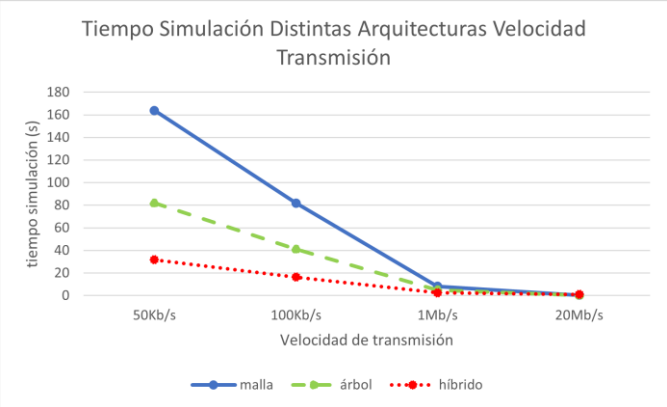


*Figura 57 Comparativa de la tres Topologías del Tiempo simulación según se aumenta la velocidad de transmisión*

# 7. Conclusiones

El presente trabajo de final de grado ha presentado un estudio teórico de las redes P2P y sus diferentes topologías, además de la simulación de tres de las posibles arquitecturas a seguir a través del entorno de simulación OMNeT++.

Para la realización de dicha simulación se han definido una serie de escenarios de pruebas, basados en dos variables principales; el número de nodos que forman la red P2P y la velocidad de transmisión de los canales de comunicación entre los distintos nodos. Durante dichas simulaciones, se ha estudiado la respuesta y adaptación de la red al aumento de nodos o velocidad de trasmisión.

La primera red P2P estudiada ha sido aquella que sigue una arquitectura descentralizada, empleando una topología de malla, en la que se ha podido comprobar que es la más afectada en cuanto al aumento de nodos de la red, en la que se podía apreciar una aumento importante tanto en el número de paquetes enviados en la red, también, como se iban descartando más paquete según el tamaño de la red va en aumento a la par que el número de saltos medios que debe realizar un nodo para cumplir la petición asignada. Todo esto, corrobora lo estudiado, en el apartado teórico, donde este tipo de red era la menos escalable, pero, no es débil a un ataque de *Single Point Failure.*

La segunda red estudiada, ha sido la red que emplea una arquitectura descentralizada con una topología jerárquica siguiendo el modelo de árbol, en la que se ha comprobado ser más eficiente que la primera red descrita, también que, al principio de la red, cuando tan solo el nodo raíz tiene la información, queda algo vulnerable. Viendo los resultados obtenidos antes, es eficiente en cuanto al número de paquetes en la red gracias a su jerarquía, ya que no inunda la red de mensajes.

Y para finalizar, la última red simulada ha sido la red que sigue la arquitectura híbrida. Esta, a la par que la red de árbol es más eficiente que la red que sigue la topología de malla al no inundar la red de mensajes, y por lo tanto la hace más escalable. Pero al igual que la red de árbol hay un punto en el que ambas redes, se estabilizan y el número de mensajes necesarios para realizar una petición y

obtener respuesta se reduce bastante en comparación a la red malla. La desventaja de este tipo de redes es tener que depender de un nodo o nodos centrales que son los súper nodos ya que son los encargados de rutear la red y sin ellos la red no puede funcionar, es por ello por lo que esta red es débil a los *Single Point Failure.*

En conclusión, dependiendo de diversos factores, interesará más desarrollar una red u otra. Si la red es pequeña y no se desea emplear una gran complejidad, la mejor solución es emplear una red descentralizada siguiendo la topología de malla, mientras que, si no se quiere depender de uno o varios nodos centrales que hagan vulnerable a tu red a un *Single Point Failure*, la mejor opción es la red descentralizada jerárquica de árbol. Mientras que si el mayor requisito buscado es la escalabilidad la mejor opción es emplear redes híbridas

# 8. Bibliografía


R. R. Schollmeier, “A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications,” *Proceedings First International Conference on Peer-to-Peer Computing*, 2001, pp. 101-102, doi: 10.1109/P2P.2001.990434.

S. Shakirat, “Client-Server Model” IOSR Journal of Computer Engineering, 2014, 16. 57-71, 10.9790/0661-16195771.

G. Gilbert Herd. (2000). The Client/Server Architecture, Server Management.

P. Peter Duchessi, InduShobha Chengalur-Smith, “Client/Server benefits, problems, best practices”, 1998, pp. 87-94, doi: 10.1145/274946.274961

B. Bruno Toledano. (5 de octubre de 2021) *Los culpables de la caída de Facebook, Whatsapp e Instagram: el BGP y las DNS.* *https://www.elmundo.es/tecnologia/2021/10/05/615c1d92fc6c8324028b45df.html*

M. Mishal Roor. (12 de noviembre de 2020). *5 Advantages and Disadvantages of Client Server Network | Drawbacks & Benefits of Client Server Network* *https://www.hitechwhizz.com/2020/11/5-advantages-and-disadvantages-drawbacks-benefits-of-client-server-network.html*

G. S. Hura, "Client-server computing architecture: an efficient paradigm for project management," *Proceedings for Operating Research and the Management Sciences*, 1995, pp. 146-152, doi: 10.1109/IEMC.1995.523924.

S. Svobodova Liba. “Client/server model of distributed processing.” *Kommunikation in verteilten Systemen I*. Springer, Berlin, Heidelberg, 1985. p. 485-498.

G. G. Fox, "Peer-to-peer networks," in *Computing in Science & Engineering*, vol. 3, no. 3, pp. 75-77, May-June 2001, doi: 10.1109/5992.919270.

X. Xiangying Yang, G. de Veciana, "Service capacity of peer to peer networks," *IEEE INFOCOM 2004*, 2004, pp. 2242-2252 vol.4, doi: 10.1109/INFCOM.2004.1354647.

R. Robin Jan Maly, “Comparison of centralized (client-server) and decentralized (peer-to-peer) networking.” *Semester thesis, ETH Zurich, Zurich, Switzerland*, 2003, p. 1-12.

R. Runhua and Shun Zhang, “An Efficient Authenticated Key Transfer Scheme in Client-Server Networks.” *Journal of Physics: Conference Series*, 2017, doi: 10.1088/1742-6596/910/1/012065.

M. M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," *Proceedings First International Conference on Peer-to-Peer Computing*, 2001, pp. 99-100, doi: 10.1109/P2P.2001.990433.

S. Siu Man Lui, Sai Ho Kwok, “Interoperability of peer-to-peer file sharing protocols.” *SIGecom Exch, 2002, pp. 25-33, doi: 10.1145/844339.844350.*

B. B. Fan, D. Chiu, J. C. s. Lui, "The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design," *Proceedings of the 2006 IEEE International Conference on Network Protocols*, 2006, pp. 239-248, doi: 10.1109/ICNP.2006.320217.

Sergi Delgado-Segura, Cristina Pérez-Solà, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Joan Borrell, "Cryptocurrency Networks: A New P2P Paradigm", *Mobile Information Systems, vol. 2018*, 2018, doi: 10.1155/2018/2159082.

Wehrle, Klaus, Mesut Günes, and James Gross, eds. *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.

Vu, Quang Hieu, Mihai Lupu, and Beng Chin Ooi. *Peer-to-peer computing: Principles and applications*. Heidelberg: Springer, 2010.

# Anexo

## Objetivos de Desarrollo Sostenible (ODS)

| Objetivos de Desarrollo Sostenibles | Alto | Medio | Bajo | No Proceed |
|---|---|---|---|---|
| ODS 1. **Fin de la pobreza.** | | | | X |
| ODS 2. **Hambre cero.** | | | | X |
| ODS 3. **Salud y bienestar.** | | | | X |
| ODS 4. **Educación de calidad.** | | | | X |
| ODS 5. **Igualdad de género.** | | | | X |
| ODS 6. **Agua limpia y saneamiento.** | | | | X |
| ODS 7. **Energía asequible y no contaminante.** | | X | | |
| ODS 8. **Trabajo decente y crecimiento económico.** | | | | X |
| ODS 9. **Industria, innovación e infraestructuras.** | | X | | |
| ODS 10. **Reducción de las desigualdades**. | | | | X |
| ODS 11. **Ciudades y comunidades sostenibles**. | | | | X |
| ODS 12. **Producción y consumo responsables.** | | | | X |
| ODS 13. **Acción por el clima.** | | | | X |
| ODS 14. **Vida submarina.** | | | | X |
| ODS 15. **Vida de ecosistemas terrestres.** | | | | X |
| ODS 16. **Paz, justicia e instituciones sólidas.** | | | | X |
| ODS 17. **Alianzas para lograr objetivos.** | | | | X |

Los Objetivos de Desarrollo Sostenible (ODS) son unos objetivos globales que los lideres mundiales adoptaron para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible.

En este TFG se tratarán 2 de los 15 objetivos descritos.

1. **Energía asequible y no contaminante**

En cuanto a la energía asequible y no contaminante, las redes P2P pueden ser unas grandes aliadas del medioambiente, ya que a diferencia de las redes Clientes-Servidor estas no necesitan de servidores centralizados para ofrecer el servicio ofertado. Hay que tener en cuenta que estos servidores suelen estar en centros de datos los cuales consumen una gran cantidad de electricidad y además la temperatura de estas salas está controlada con el fin de evitar un sobrecalentamiento de estos, todo esto supone un consumo extra energético. Todo esto con las redes P2P no ocurre, ya que los servidores centrales dejan de existir y son los propios clientes de la red los que hacen de servidores.

Gracias a esto ya no es necesario el consumo energético del que hacían uso los centros de datos donde están alojados los servidores en las redes Cliente-Servidor, por lo tanto, se reduciría la huella de carbono energética que producen estos sistemas.

## 2. **Industria, innovación e infraestructuras**

En cuanto a la industria, innovación e infraestructuras, a diferencia de las redes Cliente-Servidor, las redes P2P hacen uso de la energía más eficientemente y se aprovecha de los clientes que hacen uso de ella. Con ello aumenta su beneficio al hacer uso de los recursos de estos para hacerlos funcionar como servidores de la red; aprovechándolos mejor y además ahorrando tanto económicamente como energéticamente porque ya no es necesario montar un centro de datos con los servidores que almacenarían la información de la red.