**Trabajo - Prioridades y Colas**

**Grupo 12**

Planificación y Simulación de Redes

Departamento de Ingeniería Telemática

Beatriz Carretero Parra

Álvaro Espejo Muñoz

Alberto Flores Pereira

Miguel Alejandro García Benítez

Índice

[1 NS3 ( Network Simulator) 3](#_Toc473047895)

[1 Escenario 3](#_Toc473047896)

[2 Objetivos 5](#_Toc473047897)

[VozIP 5](#_Toc473047898)

[Interactive Video 5](#_Toc473047899)

[3 clases de la simulación 7](#_Toc473047900)

[3.1 Simulacion.cc 7](#_Toc473047901)

[3.2 VoIP.cc 8](#_Toc473047902)

[3.3 HTTP.cc 8](#_Toc473047903)

[3.4 FTP.cc 8](#_Toc473047904)

[3.5 Observador.cc 8](#_Toc473047905)

[4 Verificación del diseño y validación de los resultados 9](#_Toc473047906)

[6 futuras mejoras 9](#_Toc473047907)

[7 Conclusiones 10](#_Toc473047908)

# NS3 ( Network Simulator)

NS3 es un simulador de redes escrito en C++, con un codigo fuente de más de 300.000 lineas con una licencia GNU General Public License v2, lo que garantiza un codigo libre. Pueda ser modificado y compartido de nuevo, pero solo de forma libre y bajo ningun concepto convertirlo en un software privativo.

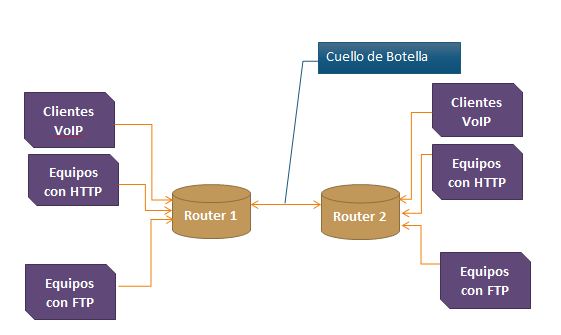
Está constituido por módulos que definen dispositivos, protocolos y utilidades reales. Es capaz de generar simulaciones de gran escala basadas en eventos discretos programados en una cola de tiempo de ejecución.

En este proyecto y debido a la necesidad de un tipo de cola **con prioridades** (***PFifoQueueDisc***) que no se encuentra disponible en la versión utilizada en clases, usaremos la versión **3-26 de ns3.**

# Escenario

El escenario planteado podría asemejarse al de dos grandes compañías operadoras, unidas por un enlace, por el cual pasa tráfico HTTP, VozIP y FTP en ambas direcciones.

Tendrá la siguiente estructura:



Como hemos mencionado anteriormente, simularemos un escenario que podría tratarse perfectamente del de dos operadoras con sus respectivos tráficos (generados por clientes) y que necesitan comunicarse entre ellas y reenvían tráfico constantemente buscando el destinatario.

Como muestra la imagen, nos interesaremos por tráfico VoIP, tráfico HTTP y tráfico FTP, ya que son tráficos que se tratan de manera distinta, porque tienen diferentes prioridades.

En el escenario nos encontramos dos routers, unidos por un enlace, que serán los encargados del reenviar el tráfico a otra LAN.

Por otro lado, a las otras interfaces de cada routers tenemos nuestros dispositivos clientes, que se encargarán de enviar tráfico y de recibirlo, teniendo como ruta por defecto el router de su LAN.

Los componentes del escenario por defecto son, por tanto:

* Clientes:
  + VoIP
  + HTTP
  + FTP
* 2 Routers conectados a través de un enlace.

Para facilitar las cosas, hemos decidido que cada cliente, también pueda actuar como servidor para garantizar la respuesta a posibles peticiones de la otra operadora.

Por tanto, cada cliente, periódicamente envía tráfico del tipo que corresponda, pero si recibe peticiones de tráfico de ese tipo, también las responderá. Esto lo hemos decidido para hacer un escenario básico y fácil de implementar.

Anteriormente, hablamos, de escenario por defecto, y es que, el escenario que muestra la foto es el escenario cuando no indicamos valores por parámetros, ya que, hemos añadido la funcionalidad de permitir elegir el número de equipos con los que queremos simular cada tráfico.

# Objetivos

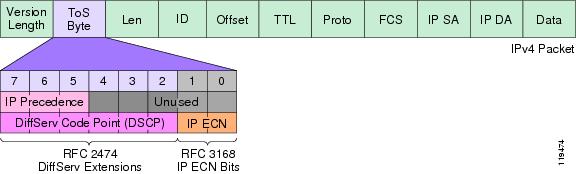
Como ya sabemos, hay grandes diferencias entre los tipos de tráfico que utilizaremos, y la mayor de ellas es la calidad de servicio de cada uno.

Por ejemplo, VozIP necesita un porcentaje de paquetes correctos igual o mayor que un 99%, al contrario que HTTP y FTP.

Es por ello que catalogaremos nuestros servicios en 3 grupos al que se le asignarán distintas prioridades según las características del tráfico:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tráfico** | **Nombre de la categoría de prioridades** | **ToS (Type of Service)** | **DSCP** |
| VozIP | Interactive Video | 184 | 46 |
| FTP | Bulk Data | 40 | 10 |
| HTTP | Best Effort | 0 | 0 |

El campo Type Of Service viaja como una etiqueta en la cabecera IP, y cuando se procesa el mensaje, se calcula su determinado DSCP.



Para poder procesar este tipo de paquetes es necesario dispositivos que estén adaptados para la lectura de etiquetas, es por ello que dotaremos de nuestros dispositivos frontera de una cola de prioridades.

Ya que nuestro escenario está orientado a una compañía operadora que ofrece servicios de este tipo, el principal objetivo de este proyecto además de medir el comportamiento del escenario según los datos de entrada (como son número de equipos, retrasos del canal, velocidad…), que explicaremos más adelante cuales son, es comprobar qué calidad de servicio de cada uno de los tráficos obtenemos variando los parámetros según la operadora requiera.

Así mismo, podríamos dar datos como la mínima velocidad del enlace para conseguir cumplir con la calidad del servicio requerida por VoIP (que es la más restrictiva con un 99% de paquetes correctos) o ver cuánto error como máximo tendría que tener el canal para cumplir con esa QoS.

Este proyecto puede de ser de gran utilidad a una gran compañía operadora, la cual necesita conocer la calidad de servicio que está ofreciendo a sus clientes a la par de tener pleno control de cuál será el comportamiento de su red en momentos de pico.

Además, estos estudios pueden permitirle a la operadora a tomar decisiones de negocio, como son las de aumentar la velocidad de un enlace o no, solucionar un problema, prepararse antes posibles subidas de tráfico o saber cómo va a reaccionar la red con el aumento de clientes abonados.

# clases de la simulación

En este proyecto como hemos mencionado anteriormente, vamos a utilizar varios tipos de servicios. A continuación, se discutirán los ficheros y clases que permitirán tanto generar, como recibir y redirigir el tráfico para conseguir nuestros objetivos.

## Simulacion.cc

Este será nuestro fichero principal contendrá la función *simulación* que se encargará de generar el escenario. Nuestra intención ha sido que el proyecto sea muy personalizable según la necesidad del quien lo ejecute. Es por ello, que el cliente puede decidir a través de la línea de comandos los siguientes elementos:

* **Tipo de gráfica**: Podrá elegir si quiere que se genere una gráfica que muestre el porcentaje de paquetes correctos en función del retraso del enlace punto a punto o una gráfica que muestre el porcentaje de paquetes correctos según el tamaño de la cola.
* **Número de nodos en las redes LAN**: Donde se podrá elegir el número de nodos totales, y no sólo eso, si no la cantidad de nodos que generen tráfico HTTP, VoIP y FTP que deseamos.
* **Capacidad de las LAN**: Se puede configurar las tasas de las redes, tanto la de la izquierda como la de la derecha.
* **Retardo**: Configuración del retardo de las LANs.
* **Retardos máximos y mínimos para el canal p2p**: Que será necesario para la generación de la gráfica 1.
* **Tamaño mínimo y máximo de la cola**: Para poder determinar el funcionamiento de la red, y para la generación de la gráfica 2.

Si no se configuran estos valores, se podrán los valores por defecto.

A continuación, se llamará a la función simulación que se encargará de montar el escenario (incluida la cola *PFifoQueueDisc*) según los parámetros recibidos.

Brevemente, ya que en el código se ve de manera más detallada, comentaremos la función *simulación.*

La función se encarga de montar el escenario, esto es creación del número de nodos que se especifique y de los objetos de tipo VoIP, HTTP y FTP para la emisión de datos, así como la creación de sumideros para la recepción de tráfico.

Es aquí además donde asignamos las prioridades según el tipo de servicio y donde creamos los objetos de tipo Observador y nos suscribimos a las trazas para obtener los datos que necesitaremos en las gráficas.

## VoIP.cc

Será nuestra clase para los dispositivos VozIp, que heredará de la clase *OnOffHelper.*

## HTTP.cc

Será nuestra clase para los dispositivos que envíen paquetes HTTP, que heredará de la clase *OnOffHelper.*

## FTP.cc

Esta clase se utilizará para generar tráfico FTP y recibirá los parámetros al llamar al constructor. Hereda de la clase *OnOffHelper.*

## Observador.cc

El observador nos permitirá obtener las estadísticas y datos que servirán para generar las gráficas. Las funciones de Observador se activarán con la llegada de nuevos paquetes o bien con el envío de los mismos.

# Verificación del diseño y validación de los resultados

Para verificar el diseño y comprobar como los resultados de simulación se ajustan a los valores teóricos calculados considerando parámetros como velocidad del canal, tiempo de on y off de los dispositivos y tamaños de paquetes hemos considerado los siguientes casos:

1. Los equipos están correctamente configurados y la cola está encolando los paquetes según su prioridad. Para ello hemos realizado varias simulaciones aumentando progresivamente el tamaño de la cola y vemos como los que salen siempre son los de la prioridad más alta independientemente del orden de llegada.
2. El tamaño de cola no está correctamente ajustado. Al aumentar el tiempo de simulación se comprueba que paquetes de baja prioridad no llegan porque son descartados.

Considerando los resultados de las simulaciones y la verificación del modelo podemos llegar a la conclusión de que nuestro modelo está correctamente diseñado y funciona como debería.

# Futuras mejoras

El tráfico por prioridades y el uso de colas abren un sinfín de posibilidades de implementaciones y escenarios.

Este escenario puede mejorar y adaptarse a las necesidades de un cliente en muchos aspectos:

1. En primer lugar, se podría hacer una comparativa de colas en la cual se podrían comparar la cantidad de paquetes útiles y elegir la mejor cola en consecuencia. Esta idea se planteó en un primer momento, pero fue descartada ya que ns3 no ofrece más colas prioritarias que la PFifoQueueDisc, y el hecho de usar colas como FIFO o RED supondrían que claramente no se cumpliría con las especificaciones de calidad de servicio de servicios como VoIP.
2. Añadir más tipos de tráfico: Es otra de las mejoras, posibles. En este proyecto se han elegido los tipos de tráfico más significativos para probar las prioridades. Sin embargo, también existen más tipos de tráfico como Streaming, que tiene otra prioridad y por lo tanto etiqueta de ToS distinta.
3. También se podrían medir más parámetros de calidad de servicio en los que no sólo se encuentran el porcentaje de paquetes útiles

# Conclusiones

Con este proyecto hemos tenido un primer acercamiento de cómo funcionan las colas de redes hoy en día y cómo los tráficos prioritarios siempre serán los primeros en salir de la cola.

Además, gracias a este proyecto hemos aprendido más sobre la configuración de colas y su programación en ns3 ya que en clase sólo hemos tratado con la cola FIFO.

También, hemos adquirido más conocimientos sobre los tipos de prioridades que existen, sus respectivos DSCP, y configurar su ToS en consecuencia.

Por último, hemos aprendido más sobre Wireshark ya que nunca lo habíamos usado con prioridades anteriormente. Aunque el ToS aparezca en la cabecera IP, no queríamos ir paquete por paquete comprobando su prioridad, y finalmente hayamos la manera de mostrar una columna que mostraba las prioridades y etiquetas DSCP para comprobar que los resultados eran los esperados.