Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE CARRERA

TÍTULO DEL TFC: Estudio, simulación y análisis del estándar IEEE 802.3az (Energy Efficient Ethernet)

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Guillermo Gil Aleixandre

DIRECTOR: David Rincón Rivera

FECHA: 24 de julio de 2012

Título: Estudio, simulación y análisis del estándar IEEE 802.3az (Energy Efficient Ethernet)

Autor: Guillermo Gil Aleixandre

Director: David Rincón Rivera

Fecha: 24 de julio de 2012

Resumen

En los últimos años las redes de telecomunicaciones se han desarrollado hasta un punto nunca visto anteriormente. Esto conlleva nuevas necesidades en ámbitos que hasta el momento no habían sido considerados como el ahorro de energía y la tolerancia a delays especialmente elevados o variables. Originalmente este trabajo pretendía incluir un estudio mediante simulaciones tanto de los protocolos destinados a mejorar el ahorro de energía como los destinados a solucionar los problemas relativos a las redes susceptibles de tener delays elevados.

Los protocolos propuestos para estos dos casos son el IEEE 802.3az, más conocido como Energy Efficient Ethernet para el ahorro de energía y la arquitectura de protocolos Delay/Disruption Tolerant Network (DTN) para las redes que presentan delays elevados. Sin embargo la carencia de simuladores para DTN que cumplieran con las necesidades del trabajo nos ha obligado a centrarnos únicamente en el estudio del estándar IEEE 802.3az.

Se ha partido de un trabajo anterior, en el que se analizó el estándar y se realizaron simulaciones preliminares en ns-2. En este trabajo se ha profundizado en el análisis de ciertas características del simulador, se ha estudiado el impacto de EEE sobre los protocolos de capas superiores (principalmente TCP y UDP) en diversos escenarios y se ha evaluado tanto el ahorro obtenido como el delay, jitter y disminución de la velocidad efectiva que provoca.

Como conclusiones generales extraídas del trabajo podemos afirmar que EEE cumple con su propósito ya que los efectos negativos son poco significativos e imperceptibles para el usuario para casi todos los escenarios y parámetros simulados, mientras que el ahorro de energía si es importante.

Title: Study, simulation and analysis of IEEE 802.3az (Energy Efficient Ethernet)
Author: Guillermo Gil Aleixandre

Director: David Rincón Rivera

Date: July, 24 th 2012

Overview

In recent years telecommunications networks have been developed to a degree never seen before. This implies need of research in areas that until now had not been considered as energy saving and tolerance to particularly high or variable delays. Originally, this thesis intended to be a simulation-based study of protocols proposed to solve both problems: to improve energy efficiency and for solving problems related to networks susceptible to high delays.

The proposed protocols for these two cases are IEEE 802.3az, better known as Energy Efficient Ethernet (EEE) for saving energy, and Delay / Disruption Tolerant Network (DTN) protocol architecture for networks that have high delays. However the lack of appropriate simulators for DTN has forced us to focus only on the study of IEEE 802.3az.

We started from a previous study, which analyzed the EEE standard and performed preliminary simulations with ns-2. This work has gone farther in the analysis of certain features of the simulator. We have studied the impact of EEE on the higher layer protocols (mainly TCP and UDP) in various scenarios and evaluated both, the savings obtained and the delay, jitter and the decrease in the effective speed.

The general conclusions drawn from the thesis are that EEE fulfills its purpose and that the negative effects are insignificant and imperceptible to the user for almost all simulated scenarios and parameters, but the power saving is still important.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1. ENERGY EFFICIENT ETHERNET 4

1.1. Introducción a Ethernet.....4

1.2. Energy Efficient Ethernet.....5

1.3. Funcionamiento de EEE.....7

1.4. Cálculo preliminar del ahorro teórico de consumo conseguido con EEE.....9

1.5. Impacto de EEE en capas superiores.....10

1.6. Trabajos previos en el ámbito EEE.....12

CAPÍTULO 2. METODOLOGÍA DE ANÁLISIS 14

2.1. Introducción.....14

2.2. NS-2.....14
2.2.1. Componentes de NS-2.....15

2.3. Características del módulo EEE 16

2.4. Análisis de las trazas y ficheros de resultados.....17
2.4.1. Ficheros de traza 17
2.4.2. Ficheros de porcentajes de tiempo.....18

2.5. Parámetros analizados.....19
2.5.1. Delay 19
2.5.2. Jitter.....19
2.5.3. Velocidad efectiva.....20
2.5.4. Porcentajes de tiempo en cada estado.....20
2.5.5. Idle * Velocidad efectiva.....20
2.5.6. Idle * Utilización 20

2.6. Escenarios simulados.....21
2.6.1. Un enlace TCP.....21
2.6.2. Dos enlaces TCP 21
2.6.3. Tres enlaces TCP 22
2.6.4. Un enlace UDP 22
2.6.5. Dos enlaces UDP.....23
2.6.6. Tres enlaces UDP 23
2.6.7. Un enlace con fuente de vídeo.....23

2.7. Variantes de TCP 24

2.8. Consideraciones adicionales.....25
2.8.1. Delay fijo adicional en los nodos.....25
2.8.2. Tiempo de propagación adicional.....25
2.8.3. Estado inicial.....26

2.8.4. Ajuste de la fuente de vídeo 26

CAPÍTULO 3. PRUEBAS REALIZADAS 27

3.1. TCP 27
3.1.1. Un enlace TCP 27
3.1.2. Dos enlaces TCP 35
3.1.3. Tres enlaces TCP 37

3.2. TCP con ventana limitada 40
3.2.1. Un enlace TCP 40

3.3. UDP 43
3.3.1. Un enlace UDP 43
3.3.2. Dos enlaces UDP 46
3.3.3. Tres enlaces UDP 48

3.4. Vídeo 48
3.4.1. Un enlace con fuente de vídeo 48

CAPÍTULO 4. CONCLUSIONES Y LÍNEAS FUTURAS 52

4.1. Conclusiones 52

4.2. Líneas futuras 53

4.3. Aspectos ambientales 54

GLOSARIO 55

CAPÍTULO 5. BIBLIOGRAFÍA 56

ANEXOS 59

5.1. Anexo 1. Modificaciones en el código del módulo EEE 59

5.2. Anexo 2. Gráficos adicionales 60
5.2.1. Un enlace TCP 60
5.2.2. Dos enlaces TCP 64
5.2.3. Tres enlaces TCP 68
5.2.4. Dos enlaces UDP 76
5.2.5. Tres enlaces UDP 77

5.3. Anexo 3. Código OTcl de las simulaciones 80
5.3.1. Un enlace TCP 80
5.3.2. Dos enlaces TCP 81
5.3.3. Tres enlaces TCP Escenario A 82
5.3.4. Tres enlaces TCP Escenario B 83
5.3.5. Un enlace UDP 84
5.3.6. Dos enlaces UDP 85
5.3.7. Tres enlaces UDP Escenario A 86
5.3.8. Tres enlaces UDP Escenario B 88
5.3.9. Un enlace con fuente de vídeo 89

INTRODUCCIÓN

Actualmente el ahorro de energía se está convirtiendo, ya sea por motivos económicos o de concienciación ecológica, en una prioridad importante para un amplio sector de la población. Las redes de telecomunicaciones no son ajenas a esta problemática. Aproximadamente un 2% de las emisiones de CO_2 son producidas por las tecnologías asociadas a las redes de telecomunicaciones [25]. Se estima que de este gasto energético aproximadamente la mitad se produce por los aparatos (servidores y equipos de comunicaciones) que están encendidos pero en un estado de inactividad. Por este motivo mejorar la eficiencia de las redes de telecomunicaciones es un objetivo importante a alcanzar en el futuro.

El estándar IEEE 802.3az [3], más conocido como Energy Efficient Ethernet, es una posible solución para esta problemática en las redes locales (LAN). Como veremos en detalle más adelante, la principal característica de este estándar es que reduce sensiblemente el consumo de las interfaces Ethernet durante el tiempo de inactividad.

Originalmente este trabajo pretendía incluir un estudio mediante simulaciones tanto de los protocolos destinados a mejorar el ahorro de energía como los destinados a solucionar los problemas relativos a las redes susceptibles de tener disrupciones o retardos elevados. Estas dos áreas son objeto de interés en la actualidad, y hay mucha actividad de investigación y desarrollo en ambas.

1) Los protocolos destinados a mejorar el ahorro de energía tienen como objetivo una reducción del consumo eléctrico de los aparatos de red ya sea haciendo entrar las maquinas un estado de consumo reducido o analizando la red para lograr adecuar el numero de aparatos en funcionamiento al tráfico de esta en cada momento.

2) Los protocolos para redes tolerantes al retardo se centran en la idea de que para determinados escenarios como transmisiones móviles, a muy larga distancia o transmisiones en los que el estado del canal sea muy variable en el tiempo los paquetes pueden tener que esperar mucho tiempo antes de ser transmitidos correctamente. Esto implica unas características específicas en el enrutamiento así como necesidades adicionales de almacenamiento en los nodos.

Hay un punto de contacto entre ellas, que queríamos explorar: si un enlace o un nodo dejan de estar disponibles, ya sea por ahorro energético o por variación de las condiciones de la red, se produce un retardo elevado o disrupción del enlace.

Los protocolos propuestos para estos dos casos son el IEEE 802.3az, más conocido como Energy Efficient Ethernet para el ahorro de energía y la arquitectura de protocolos Delay/Disruption Tolerant Network (DTN) [19] para las redes que presentan delays elevados.

Sin embargo, dadas las dificultades para atacar el área DTN, se decidió reorientar el trabajo. Analizamos los primeros simuladores implementados para DTN, algunos de ellos basados en java [21] [22], e incluso módulos para ns-2 [23] y ns-3 [20]. Sin embargo ninguno de estos simuladores para DTN se consideraron adecuados a las necesidades del trabajo, lo cual ha obligado a centrarnos únicamente en el estudio del estándar IEEE 802.3az.

Se ha partido de un trabajo anterior, en el que se analizó el estándar y se realizaron simulaciones preliminares en ns-2 [1]. El trabajo mencionado se centra en pruebas sencillas, principalmente sobre un solo enlace Energy Efficient Ethernet con una fuente de datos constante sobre TCP y en la comparación del comportamiento de los diferentes estándares que implementan EEE, 10GBASE-T, 1000BASE-T y 100BASE-TX. Se concluyó que para los escenarios simulados Energy Efficient Ethernet cumplía su propósito y los efectos negativos que introducía eran mínimos para retardos de extremo a extremo del orden de milisegundos. También se comprobó que la tendencia general del protocolo era muy similar para los diferentes estándares simulados.

En nuestro trabajo se ha profundizado en el análisis de ciertas características del simulador, se ha estudiado el impacto de EEE sobre los protocolos de capas superiores (principalmente TCP y UDP) en diversos escenarios y se ha evaluado tanto el ahorro obtenido como el delay, jitter y disminución de la velocidad efectiva que provoca.

Se han probado escenarios con varios tramos de LAN Energy Efficient Ethernet y escenarios que alternan tramos con Energy Efficient Ethernet y tramos con Ethernet clásico. Se ha estudiado el efecto de Energy Efficient Ethernet sobre trasmisiones TCP en las cuales el tamaño máximo de la ventana de transmisión era muy elevado y comparado con otras con una ventana limitada. También se ha analizado el efecto de Energy Efficient Ethernet sobre tráfico de vídeo, con sus peculiares características y requerimientos.

Como se verá en detalle más adelante, las principales conclusiones del estudio son que utilizando EEE es fácil obtener un ahorro de energía importante en entornos de uso habitual como las redes locales de usuarios domésticos con pocos efectos sobre el tráfico. El incremento del jitter y del delay que genera se imperceptible para el usuario, ya que con delays extremo a extremo grandes (a partir de un 1ms), la diferencia de prestaciones de la red es despreciable, pero el ahorro de energía sí que es significativo.

El resto de la memoria está organizado de la siguiente forma. En el capítulo 1 se explican los mecanismos que incorpora el estándar IEEE 802.3az y se analiza su potencial de ahorro teórico.

En el capítulo 2 se exponen las características de las herramientas utilizadas para la realización del trabajo así como la metodología seguida, se explican las medidas tomadas, los escenarios estudiados y las consideraciones a la hora de analizar los resultados.

En el capítulo 3 se exponen los resultados de las pruebas realizadas, se analiza el ahorro potencial de energía en cada escenario, y se estudian los condicionantes, límites e implicaciones de la aplicación de este estándar sobre el tráfico de redes Ethernet: jitter, delay y velocidad efectiva.

La memoria finaliza con las conclusiones y las líneas futuras a desarrollar en próximos estudios.

CAPÍTULO 1. ENERGY EFFICIENT ETHERNET

1.1. Introducción a Ethernet

Ethernet [28] es actualmente la tecnología dominante en redes de área local. Aunque no siempre ha sido así, los estándares modernos de Ethernet (desde Fast Ethernet a 100 Mbit/s hasta los más recientes 10 GigaEthernet y los nuevos 40GigaEth y 100GigaEth) definen una interfaz de comunicaciones en la que se envían tramas en modo serie (es decir, bit a bit) en modo full dúplex (bidireccional) entre dos nodos adyacentes. Los medios físicos son el cable de par trenzado de cobre y la fibra óptica. A modo de resumen, la tabla 1.1 presenta un resumen de los estándares y medios físicos más habituales.

Tabla 1.1. Estándares Ethernet y medios físicos más habituales [28]

Tecnología	Velocidad	Tipo de cable	Distancia máxima	Topología
10Base2	10 Mbps	Coaxial	185 m	Bus (Conector T)
10BaseT	10 Mbps	Par Trenzado	100 m	Estrella (Hub o Switch)
100BaseTX	100 Mbps	Par trenzado (categoría 5)	100 m	Estrella. Half Duplex (Hub) y Full Duplex (Switch)
100BaseFX	100 Mbps	Fibra óptica	2000 m	Estrella. Full Duplex (Switch)
1000BaseT	1 Gbps	Par trenzado (categoría 5e o 6)	100 m	Estrella. Full Duplex (Switch)
1000BaseSX	1 Gbps	Fibra óptica (multimodo)	550 m	Estrella. Full Duplex (Switch)
1000BaseLX	1 Gbps	Fibra óptica (monomodo)	5000 m	Estrella. Full Duplex (Switch)

La trama Ethernet se compone de los siguientes campos:

Fig.1.1 Campos de la trama Ethernet [30]

Preámbulo: Sirve para que el receptor detecte la transmisión y que este se sincronice con la trama.

SFD: Start Frame Delimiter. Indica el comienzo de la trama.

Dirección de Destino: Dirección MAC de destino.

Dirección de Origen: Dirección MAC de origen.

Tipo: Indica con qué protocolo de capa superior están encapsulados los datos.

Datos: Datos que incluyen cabeceras de protocolos de niveles superiores.

CRC: Comprobación de Redundancia Cíclica. Comprueba si la trama ha sido recibida sin errores.

En las simulaciones y cálculos que se presentan en este trabajo se han considerado todos los campos mencionados de la trama Ethernet, incluido el preámbulo y el delimitador de inicio de trama. A pesar de que el payload es de 1500 bytes también existen tramas denominadas jumbo frames con un payload de 9000 bytes para los estándares de alta velocidad, a partir de 1 Gbps.

Pese a su éxito comercial, Ethernet es una tecnología altamente ineficiente en el consumo de energía eléctrica. Eso es debido a que las redes de área local generalmente tienen un porcentaje de utilización inferior al diez por ciento. A pesar de ello, en muchas de las versiones de Ethernet (por ejemplo, algunas de las interfaces físicas de Fast Ethernet, y todas las de GigabitEthernet y 10GEthernet) por razones de sincronía se mantiene la línea ocupada siempre, independientemente de si se transmiten datos útiles o no. Esto provoca que el consumo eléctrico se mantenga constante y sea el máximo.

Fig. 1.2 Actividad de un enlace Ethernet con mantenimiento de sincronía.

1.2. Energy Efficient Ethernet

El estándar IEEE 802.3az [3], mas conocido como Energy Efficient Ethernet [27] tiene como objetivo la reducción del consumo eléctrico en las redes Ethernet minimizando las limitaciones y los efectos significativos sobre el tráfico de datos. Esto se logra dotando a Ethernet de mecanismos adicionales de ahorro de energía.

Ethernet divide su estándar en la especificación de dos capas principales con sus correspondientes subcapas: la capa física y la capa de enlace. La capa física proporciona los medios para transmitir bit a bit sobre el enlace de datos físico conectado a los nodos de la red. La capa de enlace es la responsable de que los datos fluyan libres de errores entre dos maquinas que estén

conectadas directamente entre si, utiliza los servicios proporcionados por la capa física.

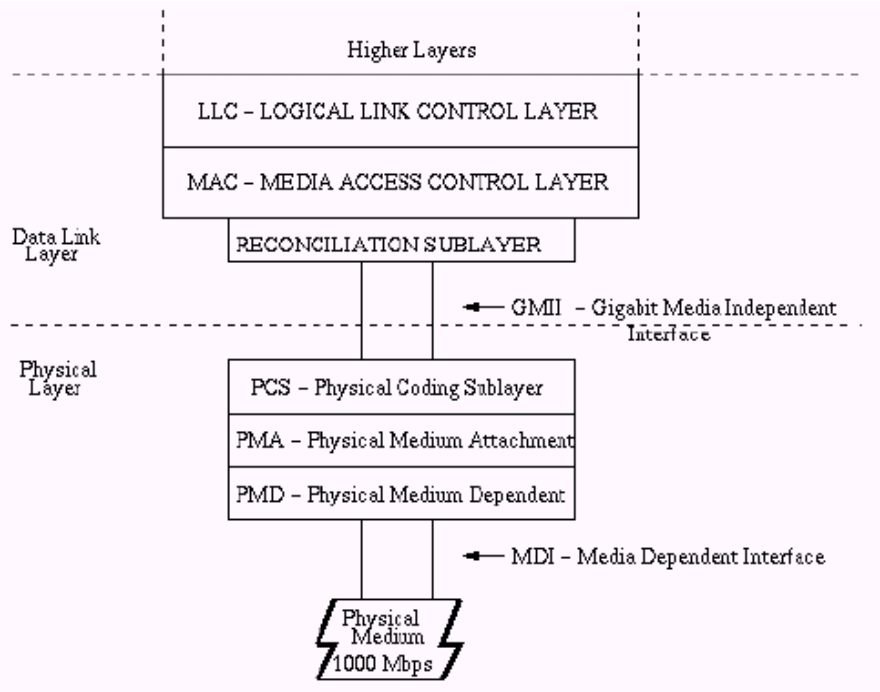

Fig. 1.3 Ejemplo de arquitectura de capas para Gigabit Ethernet [31]

En los inicios del desarrollo del estándar EEE hubo dos tendencias diferentes [25]:

- **Modificar la capa física.** Consistiría en un mecanismo de adaptación rápida de la tasa de transmisión de datos de la tarjeta que permitiría pasar dinámicamente de los diferentes estándares de 100 Mbps, 1 Gbps y 10 Gbps en función de la carga del enlace debido a que durante una transmisión a 100 Mbps la tarjeta de red tiene un consumo sensiblemente menor al que tiene para tasas superiores. Esta opción se descartó tanto debido a que no era compatible con dispositivos anteriores y por tanto su despliegue resultaría demasiado lento como a que pronto empezaron a salir al mercado los primeros dispositivos que implementaban el mecanismo LPI, descrito a continuación.

- **Modo Low Power Idle (LPI).** Esta opción consiste en introducir estados adicionales entre los que se encuentra el LPI que permite un ahorro de energía importante. Concretamente, LPI permite hacer paralelismo con lo que hacen los teléfonos móviles, que no están siempre activos, sino que “duermen” una fracción del tiempo (del orden del 90%) y se despiertan periódicamente para escuchar el canal radio y evaluar si tienen que despertarse por llamada entrante.

EEE utiliza LLDP (Link Layer Discovery Protocol) [26], un protocolo de descubrimiento de la capa de enlace especificado en el estándar IEEE

802.1ab. LLDP tiene como función que los dispositivos conectados puedan identificarse y dar a conocer su identificador y sus capacidades a los demás dispositivos de la red, incluidas las funcionalidades especificas de EEE. Es decir, permite auto negociar las funcionalidades de la tarjeta que serán utilizadas cuando ésta se conecta a la red.

1.3. Funcionamiento de EEE

EEE incorpora una serie de estados diferenciados y coordina las transiciones entre ellos. Estos estados son transparentes para los protocolos y las aplicaciones de capas superiores. Los diferentes estados del puerto en EEE son los siguientes:

- **Active**: Es el estado en que se transmiten datos de la forma habitual, es decir, en el que una trama Ethernet está siendo transmitida por el enlace.

- **Sleep**: Es estado en que se permanece durante el tiempo que el puerto necesita para pasar de Active a Low Power Idle, cuando no tiene más datos listos para transmitir.

- **Low Power Idle**: Es el estado en que se permanece durante tiempo en que el puerto no está siendo utilizado y entra en un estado de consumo eléctrico reducido, y por tanto es el estado en que se obtiene el ahorro de energía. En los primeros estudios este porcentaje de consumo se estimó en torno al 10% [6], aunque posteriormente resultados medidos en las primeras tarjetas Ethernet EEE disponibles en el mercado dieron porcentajes de consumo del 30% para 1000BASE-T y del 70% para 100BASE-TX [9].

- **Refresh**: Para mantener la sincronía entre emisor y receptor, periódicamente (a intervalos Tq) se envía una señalización Refresh para mantener el alineamiento entre transmisor y receptor. Refresh es el estado en que se permanece el tiempo que se tarda en transmitir esa señalización.

- **Wake**: Es el estado en que se permanece el tiempo que el puerto tarda en pasar del modo LPI al modo Active cuando llega una trama nueva a transmitir.

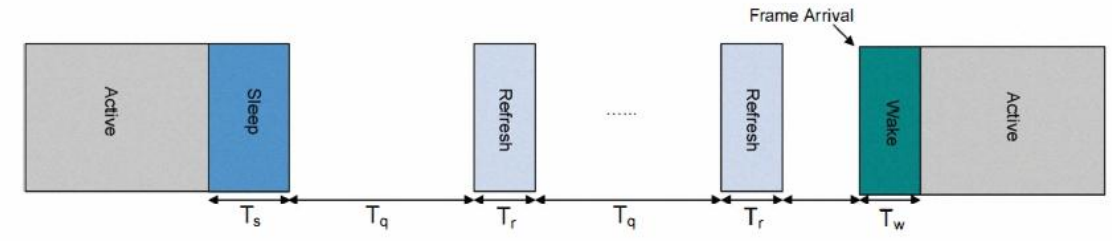

Fig. 1.4 Ejemplo de transición de estados 802.3az [7]

Si el puerto emisor se encuentra en el estado activo, continúa enviando tramas con normalidad hasta no tener ninguna pendiente de transmisión (en cola). Entonces tarda un tiempo Ts en pasar al estado de bajo consumo LPI, en el que se produce el ahorro de energía. Una vez en estado LPI, si transcurrido un tiempo Tq no hay ninguna trama nueva lista para ser transmitida entra en el estado Refresh durante un periodo de tiempo Tr, tras el que vuelve de nuevo a LPI. Si se recibe una trama nueva en el estado LPI el puerto entra en estado Wake y tarda un tiempo Tw en pasar de nuevo al estado activo y empezar la transmisión de la trama.

Esto se ve de forma más clara en la figura 1.5, que muestra el diagrama de estados del estándar IEEE 802.3az.

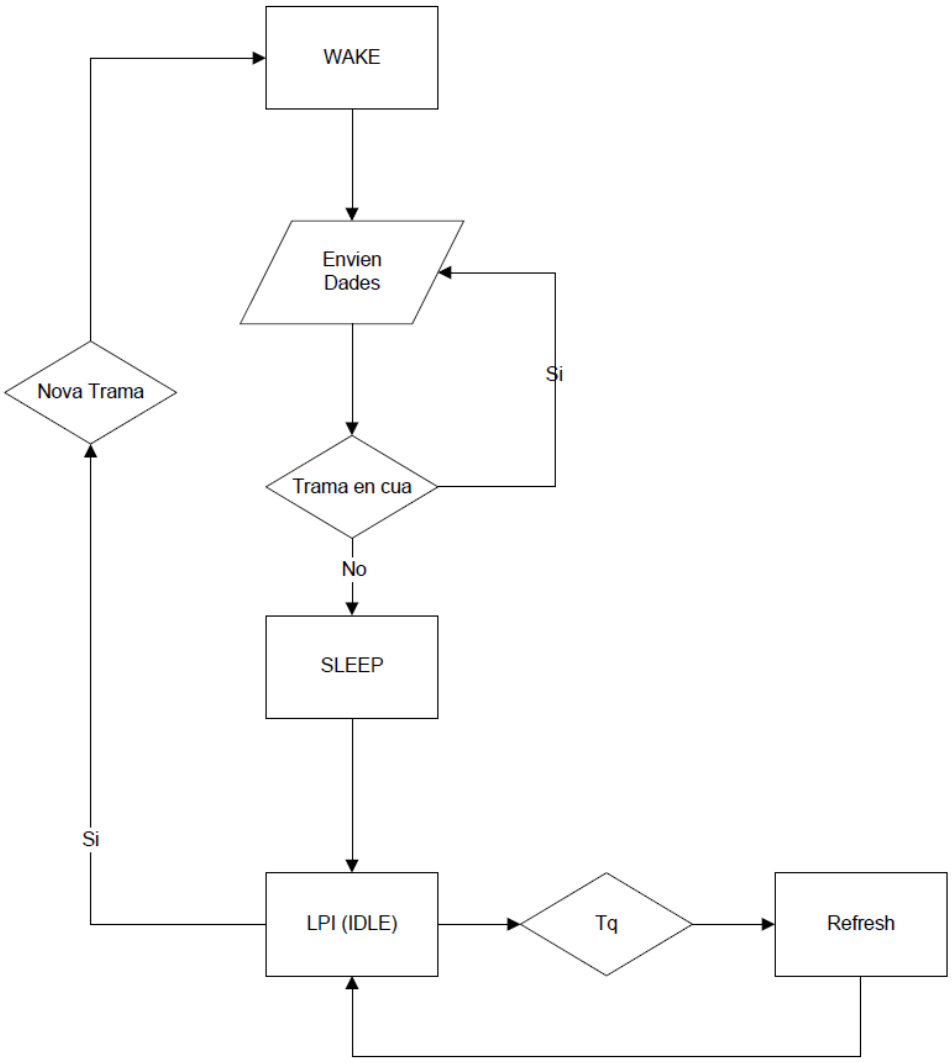

Fig. 1.5 Diagrama de estados 802.3az [1]

Como muestra la tabla 1.2, los valores de Tw, Ts, Tq y Tr varían en función de la tecnología Ethernet utilizada. A pesar de que las velocidades de Ethernet

crecen en un factor de 10, los tiempos de los parámetros EEE no siguen la misma progresión, por motivos tecnológicos. Los tiempos que tienen una mayor influencia sobre el funcionamiento de la red son Tw y Ts. Se observa que para 100BASE-TX y para 10GBASE-T son del mismo orden de magnitud que los tiempos de transmisión para MTU de 1500 y 9000 bytes, incluso un orden de magnitud menor en el caso de Tw para 100BASE-TX. En 1000BASE-T en cambio Ts es un orden de magnitud superior a los tiempos de transmisión.

Esto tiene influencia en el rendimiento de las diferentes tecnologías. Sin embargo este efecto ha sido estudiado en trabajos anteriores [1] y en este nos centraremos en el análisis de otros aspectos, tomando como referencia la tecnología 1000BASE-T.

Tabla 1.2. Valores de Tw, Ts, Tq y Tr [5]

Tecnología	Tw (µs)	Ts (µs)	Tq (µs)	Tr (µs)	Ttx MTU 1500B (µs)	Ttx MTU 9000B (µs)
10GBASE-T	4,48	2,88	39,70	1,28	1,22	7,22
1000BASE-T	16,50	182	20000	198	12,21	72,21
100BASE-TX	30	200	20000	200	122,08	722,08

1.4. Cálculo preliminar del ahorro teórico de consumo conseguido con EEE

El único estado en el que se produce ahorro de energía es el estado LPI, y por tanto el ahorro de energía obtenido con el uso de EEE es proporcional al tiempo en que el puerto se encuentra en el estado LPI. Esto se expresa en la ecuación 1.1 que relaciona el porcentaje de ahorro con el tiempo en que el puerto se encuentra en estado LPI.

$$\%Ahorro = (100 - \frac{(ConsLPI * \%LPI) + (ConsMax * (100 - \%LPI))}{ConsMax}) \quad \textbf{(1.1)}$$

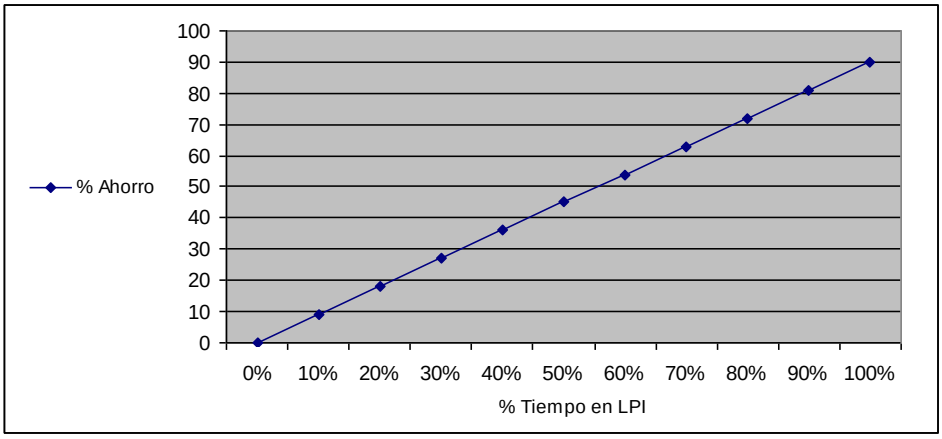

Fig. 1.6 % de ahorro en función del tiempo en LPI

Siguiendo el criterio utilizado en [6] asociamos el valor del consumo eléctrico del estado LPI en un 10% del valor máximo de consumo eléctrico. Idealmente el consumo del puerto durante el estado LPI debería ser de 0, pero por motivos tecnológicos no es posible llevar esto a la práctica.

Es fácil ver que el ahorro nunca llegará al 100% aunque el puerto no sea utilizado en ningún momento. Sin embargo dado el bajo porcentaje de utilización de las redes de área local es fácil conseguir ahorros superiores al 50%.

1.5. Impacto de EEE en capas superiores

Existen otros protocolos por encima de Ethernet en las capas del modelo de referencia TCP/IP, estándar de facto actualmente implementado de forma mayoritaria. La dinámica de estos protocolos puede verse afectada por la interacción con EEE. La figura 1.7 muestra el modelo de capas de la arquitectura TCP/IP, donde Ethernet ocuparía las dos capas inferiores (niveles Físico y Enlace), mientras que IP ocupa la capa 3 (Red). IP no influencia demasiado, ya que ofrece un servicio de transferencia de datagramas (paquetes independientes) y no introduce ninguna dinámica adicional, que sí se presenta en las capas de Transporte y Aplicación.

Fig. 1.7 Modelo de capas de la arquitectura TCP/IP

En el nivel de Transporte solemos encontrar dos protocolos:

a) **TCP** (Transmission Control Protocol). Implementa mecanismos de control de congestión basados en el reconocimiento de los paquetes tras llegar el nodo destinatario de los datos de la transmisión. Establece una ventana de congestión que va aumentando a medida que llegan los reconocimientos de paquetes que han sido transmitidos correctamente. Estos mecanismos se ven afectados por el Round-Trip-Time (RTT), el tiempo en que un paquete tarda en llegar del nodo de origen al de destino y ser confirmado. Como podemos ver, la dinámica de TCP es bastante compleja, y los incrementos y variaciones del RTT que genera EEE pueden tener efectos importantes sobre ella.

b) **UDP** (User Datagram Protocol). A pesar de que UDP no implementa mecanismos de control de congestión los datos que transporta este protocolo también son susceptibles de ser afectados por EEE. Principalmente debido a la variación del retardo de extremo a extremo de la transmisión (jitter) generado por los tiempos de wake y sleep, esta variación puede llegar a ser significativa para aplicaciones de tiempo real (como la transmisión de audio o vídeo, en servicios como Telefonía IP o IP TV), en las que es importante tanto que la transferencia extremo a extremo del paquete se realice en un tiempo acotado, como que se mantenga el patrón temporal de tráfico con las mínimas variaciones posibles (es decir, que la separación entre paquetes no se vea afectada por los efectos de la red y de los protocolos).

Además, el patrón de tráfico generado por las distintas aplicaciones (capa más alta del modelo TCP/IP) tiene su propia dinámica. Por ejemplo, la transferencia de un fichero mediante FTP tiende a adoptar la máxima velocidad de transmisión posible mientras dura la transferencia, mientras que la transmisión de audio presenta un patrón cercano a una tasa constante con una generación de paquetes de forma periódica pero que por contrapartida son sensibles al retardo y el jitter. Sin embargo existen otros patrones de tráfico más complejos

como la transmisión de vídeo, que genera ráfagas instantáneas de decenas o centenares de paquetes por cada imagen, y en el que la tasa binaria varía en función de si se va a transmitir imágenes codificadas íntegramente (I) o compensadas en movimiento (P o B, que dependen a su vez de las I), como muestra la figura 1.8.

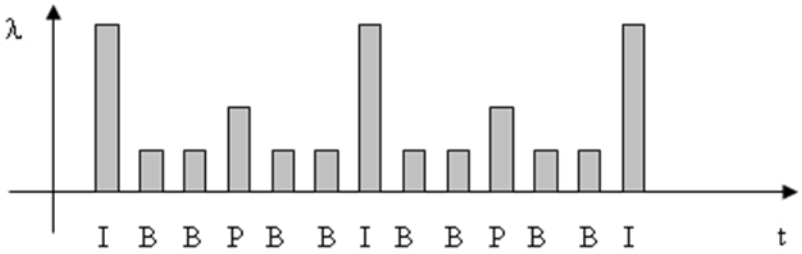

Fig. 1.8 Ejemplo de patrón de generación de datos de vídeo MPEG

1.6. Trabajos previos en el ámbito EEE

A pesar de su reciente estandarización (septiembre de 2010) existen ya una serie de artículos y trabajos previos que estudian el rendimiento de las redes EEE, muchos de los cuales fueron publicados durante su proceso de estandarización. El más influyente en este trabajo fin de carrera son los estudios de un grupo de investigación de la Universidad Antonio de Nebrija (Madrid) que nos ha cedido el código de su módulo de simulación EEE para ns-2. Este módulo será descrito en detalle en el capítulo 2. A continuación destacamos los trabajos y resultados más importantes.

En el articulo "Performance Evaluation of Energy Efficient Ethernet" [6] de septiembre de 2009 se establece de forma teórica la curva Consumo de energía – carga en 100Mbps, 1Gbps y 10Gbps utilizando la siguiente expresión.

$$P = P_{lowpower}(1 - load) + P_{active} * load \qquad \textbf{(1.2)}$$

En el siguiente articulo “Increasing the MTU size for Energy Efficiency in Ethernet” de julio de 2010 [7] se estudia en un entorno simulado en ns-2 los efectos de utilizar tramas de 9000 bytes (Jumbo Frames), y concluye que al producirse menos transmisiones mejora la eficiencia y por tanto el consumo seria menor.

En el articulo “Burst Transmisión in Energy Efficient Ethernet” de Julio de 2010 [8] se experimenta con el uso de la “coalescencia”. Esta técnica consiste en el uso de buffers en los que se almacenan las tramas para enviarse posteriormente en intervalos definidos.

En el articulo “The Road to Energy Efficient Ethernet” de noviembre de 2010 [4] se realizan una serie de pruebas con TCP y FTP, con EEE y la técnica de coalescencia sobre un entorno simulado en ns-2.

El articulo “An Inicial Evaluation of Energy Efficient Ethernet” de mayo de 2011 [9] es el primero que utiliza una interfaz real que implementa el Draft 3.2 del estándar EEE (Realtek RTL8111E). En este artículo se caracteriza el consumo eléctrico de las tarjetas Ethernet con tráfico y sin tráfico con una transferencia de datos sobre TCP y FTP entre dos ordenadores, se mide el consumo con envíos de pings de 250 bytes y se caracteriza la curva que relaciona consumo de energía con carga de enlace. El consumo de las tarjetas Ethernet medido en ausencia de tráfico (Estado LPI) es entorno al 30% del consumo máximo para 1000Base-T y entorno al 70% para 100Base-T.

Por último el TFC de la EETAC “Estudi i simulacions de l’estàndard Energy-Efficient-Ethernet” de septiembre de 2011 [1] estudia de forma similar a este trabajo los efectos de EEE sobre el tránsito y el potencial de ahorro del mismo. Se centra en pruebas más sencillas realizadas con los estándares para 10GBASE-T, 1000BASE-T y 100BASE-TX y compara los efectos de los diferentes tiempos de Wake y Sleep para cada uno de ellos sobre transferencias FTP.

Nuestro trabajo supone una continuación de [1] en la que, partiendo de la conclusión de que el efecto sobre el tráfico es similar para todas las tecnologías mencionadas nos centramos en el estándar 1000BASE-T sobre escenarios y condiciones de simulación más complejos. Concretamente, analizaremos:

a) EEE sobre uno y varios tramos de LAN en la misma transmisión de datos que implementen EEE
b) EEE sobre tramos de EEE consecutivos y no consecutivos en la misma transmisión de datos.
c) EEE sobre transmisiones FTP con ventana de transmisión de TCP de distintos tamaños.
d) EEE sobre transmisiones de tasa constante sobre UDP
e) EEE sobre transmisiones de tráfico de vídeo sobre UDP

CAPÍTULO 2. METODOLOGÍA DE ANÁLISIS

2.1. Introducción

Los escenarios para la realización de este trabajo han sido realizados en un entorno de simulación ns-2. Durante este capítulo se presenta el simulador utilizado y detallan las características del módulo adicional que implementa EEE. También se explica cada una de los parámetros analizados y el criterio utilizado para medirlos. Por último se hace una presentación de los escenarios que han sido simulados, los resultados de los cuales se presentan en el siguiente capítulo.

2.2. NS-2

En este trabajo hemos utilizado como plataforma de simulación el simulador ns-2 (Network Simulator 2) [10], uno de los simuladores de redes más extendido actualmente. Ns-2 es un simulador determinista de eventos discretos, en el que las acciones simuladas están asociadas a los eventos y no al tiempo real de simulación. Un evento en un simulador de eventos discretos consiste en una sucesión de acciones, un tiempo de ejecución y una referencia para el siguiente evento dependiente de este.

Este software es una potente herramienta capaz de simular una gran diversidad de fuentes de tráfico, topologías y protocolos de red de diferentes capas tanto para redes cableadas como inalámbricas.

Es un programa open source desarrollado en C++ sobre plataforma UNIX (Linux, Solaris, Mac OS, BSD) que también puede ser ejecutado sobre una plataforma Windows mediante Cygwin. Los escenarios de simulación se definen en OTcl [17], lo que lo hace mucho más amigable y fácil de usar que C++. El hecho de que sea un programa open source permite al usuario realizar las modificaciones que necesite en el código C++, aunque estas modificaciones puedan ser complejas. Con el paso del tiempo han ido saliendo versiones nuevas de ns-2 y una cantidad importante de módulos creados por los usuarios para simular los protocolos más diversos. Es esta variedad de módulos para ns-2 lo que lo convierte en un entorno de simulación tan popular. En nuestro caso uno de esos módulos es el desarrollado por el grupo de investigación de la Universidad Antonio de Nebrija que simula una implementación de EEE y hemos utilizado para las pruebas de este trabajo.

La versión de ns-2 utilizada para nuestras pruebas es la 2.35 publicada el 4 de noviembre de 2011, la versión más reciente hasta el momento.

2.2.1. Componentes de NS-2

El software de ns-2 esta implementado de forma modular de manera que el simulador es el núcleo y además consta de una serie de componentes complementarios:

- **ns**: El núcleo del programa y el simulador en sí.
- **nam**: Visualizador de redes, herramienta visual que permite ver animaciones de los paquetes sobre la topología durante el transcurso de la simulación. Útil para aspectos didácticos y debugado de las simulaciones, aunque no tanto para investigación y simulaciones largas.
- **Pre-procesado**: Scripts escritos en lenguaje OTcl ya sea a mano o mediante un generador de topología. Son estos scripts los que conforman los parámetros de la simulación.
- **Post-procesado**: La salida de ns-2 son los ficheros de trazas que pueden ser analizados con posterioridad por programas de gráficos como Xgraph o Tracegraph, scripts AWK o cualquier tipo de procesado. En este trabajo los datos de estos ficheros han sido analizados mediante un tratamiento con Excel.

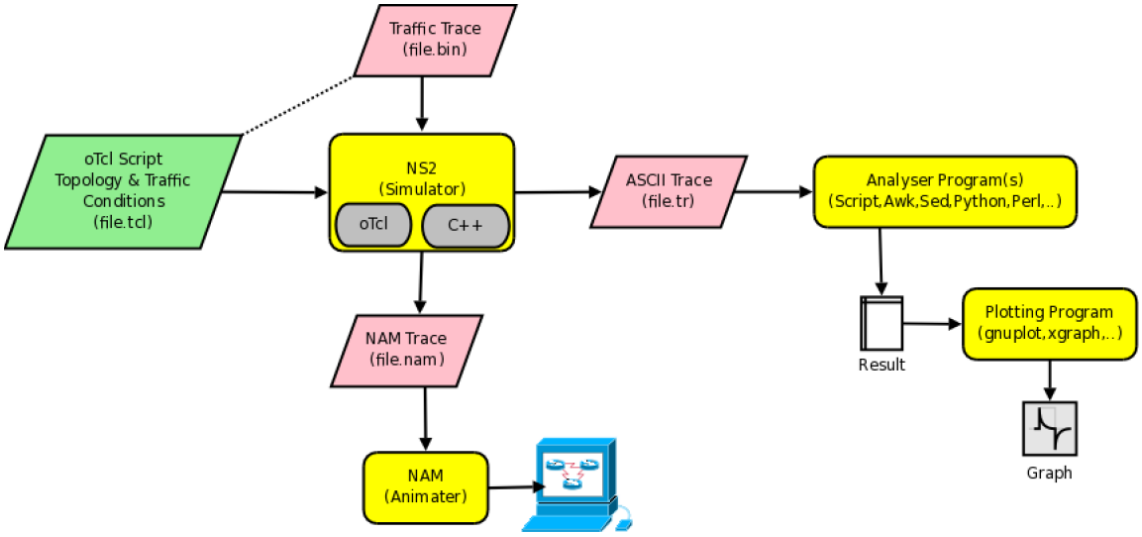

Fig.2.1 Componentes del software ns-2 [16]

Desde el punto de vista del usuario, ns-2 es un intérprete de scripts OTcl en el cual la entrada es un fichero *.tcl y la salida un fichero de trazas *.tr con los datos referentes a la simulación. Para más detalles sobre el formato de los scripts y de las trazas, se recomienda consultar el apartado 2.4 y las referencias [10] y [12]

2.3. Características del módulo EEE

La implementación actual de ns-2 no dispone de ningún módulo para simular EEE. Sin embargo los investigadores de la universidad Antonio de Nebrija Alfonso Sánchez Macian Pérez y Pedro Reviriero Vasolla crearon uno modificando la implementación de Ethernet de ns-2 [18]. Ese ha sido el módulo que se ha utilizado en nuestras simulaciones para este trabajo y por ello agradecemos a los autores el haber podido disponer de su código.

Además de una implementación del protocolo EEE, este módulo nos proporciona por pantalla el porcentaje de tiempo que cada puerto ha permanecido en cada uno de los estados posibles después de cada transición sleep. Las estadísticas que proporciona referentes a los estados son las siguientes: %Tsleep, %Twake, %Tactive, %Tidle, %Trefresh, %Ttransition, donde:

$$\%Tsleep + \%Twake + \%Tactive + \%Tidle + \%Trefresh = 100\% \qquad \textbf{(2.1)}$$

$$\%Twake + \%Tsleep = \%Ttransition \qquad \textbf{(2.2)}$$

Posteriormente este módulo fue modificado para que las impresiones fueran volcadas a ficheros en lugar de mostrarlas por pantalla. El resultado es que se generan dos ficheros para cada uno de los siguientes estados: Active, Transición, Refresh e Idle. Uno de los ficheros contiene los porcentajes de tiempo correspondientes (factive.txt, ftransition.txt, fidle.txt, frefresh.txt) y el otro la acumulación del tiempo transcurrido en cada estado (factiveT.txt, ftransitionT.txt, fidleT.txt, frefreshT.txt). También se generan dos ficheros de comprobación adicionales: TransT.txt con la suma de los tiempos de Wake y Sleep y TT.txt con la suma del tiempo acumulado contando el tiempo acumulado en cada estado.

En nuestro análisis se han utilizado los ficheros factiveT.txt, ftransitionT.txt, fidleT.txt, frefreshT.txt únicamente como comprobación y se han analizado principalmente los ficheros factive.txt, ftransition.txt, fidle.txt, frefresh.txt resultantes de cada simulación.

Después de cada simulación los ficheros generados (factiveT.txt, ftransitionT.txt, fidleT.txt, frefreshT.txt, factive.txt, ftransition.txt, fidle.txt, frefresh.txt, transT.tx, TT.txt) deben ser borrados o renombrados antes de realizar una simulación posterior en el mismo directorio. De lo contrario los nuevos resultados se imprimirían en los ficheros existentes en las líneas posteriores a las ya escritas. Aunque esto no supone una pérdida de datos, fácilmente puede dar pie a errores durante la fase de análisis.

2.4. Análisis de las trazas y ficheros de resultados

Los datos generados tras las simulaciones que se han empleado son básicamente los extraídos de los ficheros de traza y los porcentajes de tiempos del módulo EEE. A continuación se incluye un fragmento de cada uno de estos ficheros como ejemplo y se explica el significado de los datos que contiene.

2.4.1. Ficheros de traza

Los ficheros de trazas son los ficheros fundamentales para el análisis de las simulaciones en ns-2. El ejemplo de la Fig 2.2 muestra la vida de los tres primeros paquetes de una transmisión de tasa constante sobre UDP.

```
h 0.000182 0 2 cbr 1500 ------- 0 0.0 1.0 0 0
+ 0.000192 0 2 cbr 1500 ------- 0 0.0 1.0 0 0
- 0.000192 0 2 cbr 1500 ------- 0 0.0 1.0 0 0
r 0.000235 2 1 cbr 1500 ------- 0 0.0 1.0 0 0
h 0.025182 0 2 cbr 1500 ------- 0 0.0 1.0 1 1
+ 0.025192 0 2 cbr 1500 ------- 0 0.0 1.0 1 1
- 0.025192 0 2 cbr 1500 ------- 0 0.0 1.0 1 1
r 0.025235 2 1 cbr 1500 ------- 0 0.0 1.0 1 1
h 0.050182 0 2 cbr 1500 ------- 0 0.0 1.0 2 2
+ 0.050192 0 2 cbr 1500 ------- 0 0.0 1.0 2 2
- 0.050192 0 2 cbr 1500 ------- 0 0.0 1.0 2 2
r 0.050235 2 1 cbr 1500 ------- 0 0.0 1.0 2 2
```

Fig. 2.2 Ejemplo de fragmento de fichero de traza

Ns-2 modela los nodos de una LAN de la forma que indica la figura 2.3, con una cola de salida para cada nodo conectado al tramo de la LAN.

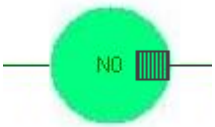

Fig. 2.3 Modelo del nodo

En la traza el primer campo indica el evento asociado al paquete: h indica el tiempo de generación del paquete en el nodo, + indica el instante en que el nodo entra en la cola de salida, - el instante en el que el nodo está listo para abandonar la cola de salida y r el instante en que es recibido por completo por el siguiente nodo. Es entre los eventos – y r cuando se cuenta el tiempo de transmisión del paquete, el tiempo de propagación y el tiempo que el paquete permanece a la espera de que el puerto pase a estado Active.

El segundo campo indica el instante en el que se produce el evento. En el ejemplo el primer paquete es generado a los 182 µs.

El tercer y cuarto campos indican los identificadores de los nodos de origen y destino para cada salto. En las redes de área local ns-2 asigna uno de estos identificadores a cada tramo de la LAN empezando por el 0 y en el orden en que son declarados en el script OTcl.

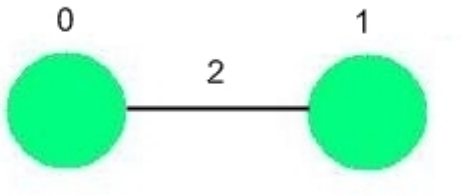

Fig. 2.4 Asignación de identificadores para el ejemplo de la figura 2.2

En nuestros scripts se declara sistemáticamente en primer lugar los nodos y en segundo lugar los tramos de la LAN que los conectan. Por tanto para el ejemplo anterior los identificadores quedan como muestra la figura 2.3, 0 para el primer nodo, 1 para el segundo nodo y 2 para el tramo de LAN que los conecta.

El quinto campo indica el tipo de datos que transporta el paquete, en este caso CBR (Constant Bit Rate) significa que es una transmisión de datos de tasa constante.

El sexto campo indica el tamaño en bytes del paquete.

Los campos del séptimo al decimotercero son diferentes flags, que al nivel al que hemos hecho nuestras simulaciones no son relevantes. La presencia de – indica que están desactivados.

El decimocuarto campo es el identificador de flujo, es decir el flujo de nivel de transporte al que pertenecen, para el ejemplo todos los paquetes pertenecen al mismo flujo que transmite datos a tasa constante.

Los campos decimoquinto y decimosexto son las direcciones de origen y destino del paquete. En el ejemplo 0.0 para el nodo 0 y 1.0 para el nodo 1.

El decimoséptimo campo es el número de secuencia y el decimoctavo es un identificador único asignado a cada paquete de la simulación, lo que nos permite seguir todos los eventos asociados a la vida de cada paquete, desde que es generado en su fuente hasta que es consumido en el receptor.

2.4.2. Ficheros de porcentajes de tiempo

Los ficheros de porcentajes de tiempo generados por el módulo EEE almacenan para cada transición al estado sleep una línea nueva con el porcentaje de tiempo que el puerto a estado en un estado determinado con respecto al tiempo total de la simulación hasta ese momento. La cifra de la

última línea del fichero es el porcentaje de tiempo que el puerto ha permanecido en ese estado en el momento de finalizar la última transición de estado sleep. Las simulaciones están ajustadas para terminar en el momento de la última transición de estado sleep. La Fig 2.5 muestra el porcentaje de tiempo en estado idle para los tiempos asociados a las 14 primeras transiciones de estado a sleep de una simulación de un enlace con UDP a tasa constante.

```
0.0000
2.4832
96.8452
97.5991
97.8531
97.9806
98.0573
98.1084
98.1450
98.1725
98.1939
98.2110
98.2250
98.2366
```

Fig. 2.5 Ejemplo de fragmento de fichero de porcentaje tiempo

2.5. Parámetros analizados

Con el fin de analizar la influencia de EEE en el tráfico de datos hemos monitorizado una serie de métricas y parámetros adecuados a cada escenario. En este apartado se enumera y explica cada una de ellas.

2.5.1. Delay

Es el retardo de transferencia de cada paquete. Hemos tomado la medida del delay de extremo a extremo, definida desde el instante en que un paquete se genera en el nodo emisor hasta que es recibido en su totalidad por el receptor final. El delay extremo a extremo está afectado por diversos factores: el retardo de cada enlace, el tiempo de transmisión, las transiciones de estado de EEE y el tiempo de espera en cola. Se expresa en unidades de tiempo.

2.5.2. Jitter

El jitter se define cualitativamente como la variación del retardo extremo a extremo (delay), y tiene importancia en el caso de transmisiones en tiempo real. Hay varias maneras de definir matemáticamente esta variación, como por ejemplo la desviación estándar del delay, pero en los protocolos de Internet se suelen usar definiciones más sencillas y fáciles de implementar en máquinas

que tienen que estimar el jitter en tiempo real. En la definición que hemos usado, siguiendo el criterio de [1], es la media de la medida de la diferencia del delay extremo a extremo de un paquete con respecto al delay extremo a extremo del paquete anterior. Siguiendo esta definición, es una medida por tanto de la variación del delay entre paquetes consecutivos. Definimos el jitter como:

$$J = E\left[\left|D_i - D_{i-1}\right|\right] \qquad \textbf{(2.3)}$$

Donde D_i es el delay asociado al paquete i-ésimo de la transmisión. Se expresa en unidades de tiempo.

2.5.3. Velocidad efectiva

La velocidad efectiva es la medida de la tasa media a la que la red transmite datos útiles sin contar las cabeceras. Evaluamos los datos a nivel de EEE; por tanto las cabeceras de protocolos de nivel superior (IP, TCP o UDP) son consideradas como parte de estos datos útiles. Se expresa en unidades de tasa o velocidad (bits/s).

2.5.4. Porcentajes de tiempo en cada estado

Es la medida del tiempo en que un puerto se encuentra en cada uno de los estados mencionados en el apartado 2.3 con respecto al tiempo total de simulación. Se mide en fracción o porcentaje.

2.5.5. Idle * Velocidad efectiva

Es la figura de merito definida en [1] que consiste en multiplicar el porcentaje de tiempo que el puerto se encuentra en estado LPI por la velocidad efectiva medida durante la simulación. Por el modo en que funciona EEE cuanto mayor sea el ahorro de energía por inactividad, menor será la velocidad efectiva. Esta medida pretende llegar a una zona de compromiso entre ambos parámetros.

2.5.6. Idle * Utilización

De forma similar a la figura de merito anterior, consiste en multiplicar la utilización del enlace por el porcentaje de tiempo que el puerto se encuentra en estado LPI. Esto nos da una medida del compromiso entre el ahorro potencial y la utilización del enlace.

2.6. Escenarios simulados

A continuación se enumeran cada uno de los escenarios simulados, se justifica su interés, y se mencionan las medidas tomadas y el efecto que se pretende contrastar.

2.6.1. Un enlace TCP

En este escenario se medirá el efecto de EEE en una conexión TCP con un único enlace EEE y una fuente FTP que proporciona datos de forma continua sobre la misma conexión TCP. Este será el escenario básico para el análisis de los efectos sobre el protocolo TCP y sobre él se medirá la velocidad efectiva, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado.

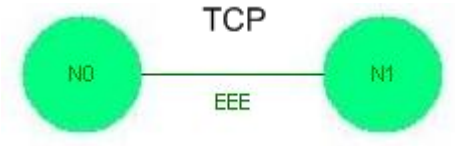

Fig. 2.6 Escenario para un enlace TCP

Tanto sobre este como sobre el resto de escenarios con el protocolo TCP se simularán dos situaciones diferenciadas, una en la que la ventana de transmisión es suficientemente grande para no limitar la velocidad y otra en que la ventana implica una limitación adicional. La ventana óptima se define acorde con la siguiente expresión.

$$Wopt = RTT * Bw \qquad \textbf{(2.4)}$$

Por tanto, el aumento del RTT producido por efecto de EEE puede afectar a la consecución de la ventana óptima durante la transmisión.

Además se simulará sobre este escenario básico diferentes versiones de TCP para comprobar el efecto de los algoritmos de control de congestión sobre cada una.

2.6.2. Dos enlaces TCP

En este escenario se medirá el efecto de EEE en una conexión TCP con dos enlaces EEE y una fuente FTP que proporciona datos de forma continua sobre la misma conexión TCP. Sobre este escenario se medirá la velocidad efectiva, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado. Este escenario está pensado para analizar el efecto del protocolo EEE sobre más de un enlace consecutivo, y evaluar el impacto de la acumulación de los efectos.

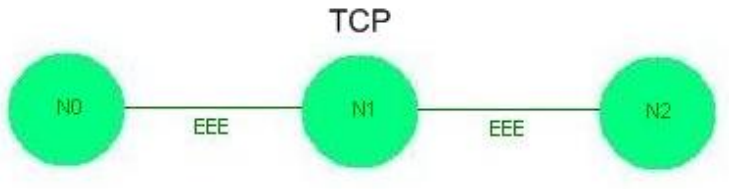

Fig. 2.7 Escenario para dos enlaces TCP

2.6.3. Tres enlaces TCP

En este escenario se medirá el efecto de EEE en una conexión TCP con dos enlaces EEE y un enlace Ethernet sin EEE, utilizando una fuente FTP que proporciona datos de forma continúa sobre la misma conexión TCP.

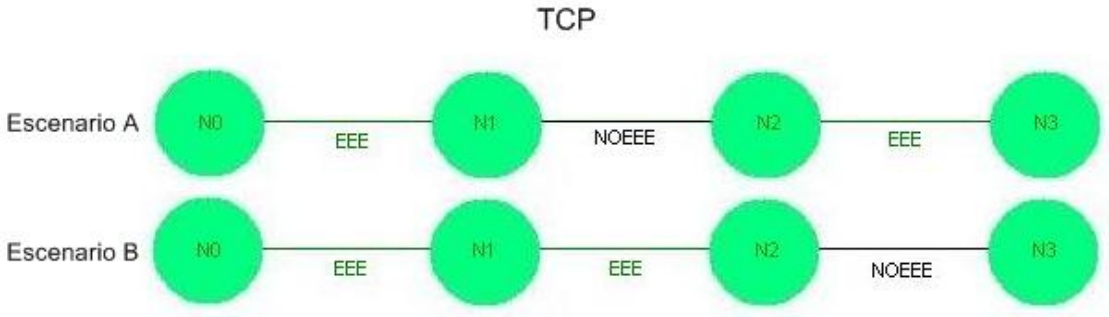

Fig. 2.8 Escenarios para tres enlaces TCP

Se simularán dos escenarios diferentes, uno en el que ambos enlaces EEE serán consecutivos y otro en el que el enlace que no es EEE se encontrará entre los otros dos. Sobre estos escenarios se medirá la velocidad efectiva, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado. Este escenario está pensado para analizar el efecto del protocolo EEE cuando interactúa con otros tramos de la misma red local que no disponen de EEE, y si el orden de los enlaces en la cadena de transmisión tiene algún tipo de influencia.

2.6.4. Un enlace UDP

En este escenario se medirá el efecto de EEE en una conexión UDP con un único enlace EEE y una fuente que proporciona datos a velocidad constante. Este será el escenario básico para el análisis de los efectos sobre el protocolo UDP y se medirá la utilización del enlace, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado.

Fig. 2.9 Escenario para un enlace UDP

2.6.5. Dos enlaces UDP

En este escenario se medirá el efecto del protocolo EEE en una conexión UDP con dos enlaces EEE, utilizando una fuente que proporciona datos a velocidad constante. Sobre este escenario se medirá la utilización del enlace, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado. Este escenario está pensado para analizar el efecto del protocolo EEE sobre más de un enlace consecutivo, pero sin introducir la dinámica de protocolos complejos como TCP.

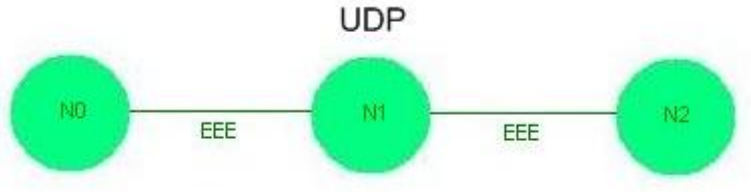

Fig. 2.10 Escenario para dos enlaces UDP

2.6.6. Tres enlaces UDP

En este escenario se medirá el efecto de EEE en una conexión UDP con dos enlaces EEE y un enlace Ethernet sin EEE, utilizando una fuente que proporciona datos a velocidad constante. Se simularán dos escenarios diferentes, uno en el que ambos enlaces EEE serán consecutivos y otro en el que el enlace que no es EEE se encontrará entre los otros dos. Sobre este escenario se medirá la utilización del enlace, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado. Este escenario está pensado para analizar el efecto del protocolo EEE cuando interactúa con otros tramos de la misma red local que no disponen de EEE, y si el orden de los enlaces en la cadena de transmisión tiene algún tipo de influencia.

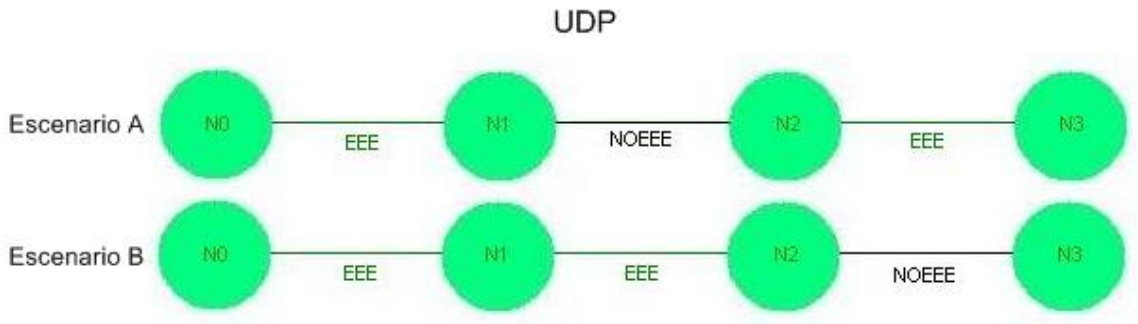

Fig. 2.11 Escenarios para tres enlaces UDP

2.6.7. Un enlace con fuente de vídeo

En este escenario se medirá el efecto de EEE en una conexión UDP con un único enlace EEE, utilizando una fuente sintética de vídeo MPEG-4. Sobre este escenario se medirá la utilización del enlace, el delay extremo a extremo, el jitter y el porcentaje de tiempo en que la interfaz permanecerá en cada estado. Este escenario está pensado para analizar el efecto del protocolo EEE sobre el protocolo UDP cuando transporta datos de fuentes con patrones de tráfico complejos como son los datos de vídeo.

Fig. 2.12 Escenario para un enlace con fuente de vídeo

2.7. Variantes de TCP

En nuestro trabajo se han realizado una serie de simulaciones analizando el protocolo TCP en las cuales se ha utilizado diferentes variantes del mismo para ver el efecto de EEE en cada una de ellas, las variantes escogidas han sido TCP-Linux [34], TCP-Reno [33] y TCP-Vegas [32]. A continuación se mencionan las características de cada una de estas variantes.

TCP-Reno: La característica principal de esta variante con respecto a su versión anterior (TCP-Tahoe) es que incluye el mecanismo de Fast Recovery (Recuperación rápida), en el cual una vez que el transmisor de datos recibe una cantidad de ACKs duplicados que supera cierto valor de umbral retransmite un paquete y reduce su ventana de congestión a la mitad, en lugar de empezar con Slow-Start. Durante la retransmisión rápida el transmisor incrementa su ventana según el número de ACK duplicados que ha recibido, bajo la suposición de que cada ACK duplicado que le llega es una indicación de que un paquete salió del transmisor y fue recibido por el receptor. Después de la recepción de un ACK correspondiente a datos nuevos, el transmisor sale de la fase de Fast Recovery. Esto implica que a diferencia de versiones anteriores al no volver a Slow-Start después de la pérdida de un solo paquete evita una reducción innecesaria de la velocidad por perdidas aisladas. No obstante en los escenarios simulados no se prevén pérdidas y por tanto no se espera una gran diferencia entre la velocidad efectiva utilizando EEE y Ethernet clásico.

TCP-Linux: Es la implementación de TCP propia de los sistemas operativos Linux. El código TCP en Linux incluye varias implementaciones entre las cuales las más habituales son TCP-newReno y CUBIC-TCP. En nuestras simulaciones hemos utilizado la implementación de TCP-newReno que incluye el mecanismo de SACK (acuse de recibo selectivo) el cual permite la identificación en los ACK de los segmentos TCP concretos que se han perdido. Del mismo modo que para TCP-Reno, en los escenarios simulados no se prevén pérdidas y por tanto no se espera una gran diferencia entre la velocidad efectiva utilizando EEE y Ethernet clásico.

TCP-Vegas: La característica principal de esta variante es que ajusta el tamaño de la ventana de congestión fundamentándose tanto en la pérdida de paquetes como en la variación del RTT de estos. TCP-Vegas mide el RTT de los paquetes transmitidos y confirmados, de modo que cuando el RTT aumenta asume que la red esta congestionada y por tanto reduce el tamaño de la ventana de congestión mientras que si el RTT disminuye asume que la red no esta congestionada y aumenta el tamaño de la ventana de congestión. Se espera que la variación del RTT ocasionada por EEE tenga algún efecto sobre esta variante de TCP. Se ha escogido esta versión debido a que su funcionamiento es similar a Compound TCP, versión de TCP de uso habitual en los sistemas operativos Windows.

2.8. Consideraciones adicionales

Durante la realización del trabajo hemos detectado una serie de efectos no deseados ocasionados por la implementación de las herramientas de simulación que se han utilizado en el trabajo. Sin embargo, estos efectos tienen una influencia limitada sobre los resultados, por lo que hemos procedido a corregirlos y mantenemos la validez de los resultados obtenidos. A continuación mencionamos los efectos adicionales y sus repercusiones en los resultados.

2.8.1. Delay fijo adicional en los nodos

Durante el análisis de las trazas generadas por NS-2 hemos detectado que en las simulaciones de redes locales los paquetes esperan antes de entrar en la cola de salida de cada nodo (entre los eventos h y + de los ficheros de traza) un tiempo igual al delay del enlace por el que los deben transmitir. Esto implica un delay adicional constante en todos los paquetes de la simulación que provoca un efecto equivalente a doblar el tiempo de propagación del enlace. Este tiempo adicional al producirse a todos los paquetes tanto para las simulaciones de tramos con EEE como a las demás y tratarse de un tiempo constante puede descontarse y no tiene repercusiones significativas sobre los resultados medidos.

2.8.2. Tiempo de propagación adicional

También se ha detectado un tiempo de propagación adicional constante en las simulaciones sobre redes locales de 4 µs que atribuimos a otros mecanismos no analizados de las redes LAN (aparentemente, el método de acceso al medio, pero no lo hemos podido confirmar). Una vez más se trata de un tiempo constante que encontramos en cada paquete que es transmitido y por lo tanto no tiene repercusiones significativas sobre los resultados medidos.

2.8.3. Estado inicial

En las simulaciones en que se utiliza EEE el estado inicial de los puertos en el instante 0 es de sleep en lugar de LPI como sería deseable, lo que provoca que al empezar la simulación transcurre un tiempo de sleep para pasar al estado de LPI. Este tiempo de sleep adicional al principio de la simulación suponía un delay no deseado que resultaba significativo en determinadas pruebas. Para evitar este efecto las simulaciones empiezan en el instante 182 µs (que corresponde al tiempo de sleep de 1000BASE-T), en lugar de en el instante 0.

2.8.4. Ajuste de la fuente de vídeo

En el caso del tráfico de vídeo se observó un comportamiento excesivamente irregular para la fuente sintética de vídeo MPEG-4. En los instantes que correspondían a una imagen codificada íntegramente se generaban al nodo emisor una serie de paquetes en el mismo instante de tiempo en lugar de generarse de forma sucesiva en una ráfaga de paquetes seguidos, separados por el tiempo de transmisión de cada uno de ellos, como correspondería a una fuente de vídeo realista. Las ráfagas instantáneas provocan pérdidas en la cola del primer nodo de la red, ya que sobrepasan la longitud de la cola, invalidando así el análisis. Se intentó modificar el tamaño de la cola de los nodos EEE, pero no fue posible. Para eliminar este efecto hemos utilizado la siguiente solución.

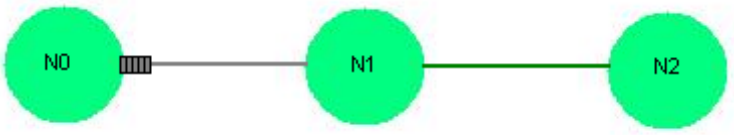

Fig. 2.13 Esquema de la solución a las ráfagas instantáneas de vídeo

La solución elegida para eliminar este efecto consiste en la adición de un nodo previo enlazado con el primer nodo de la red LAN. Este enlace previo es un enlace genérico con una cola de salida lo suficientemente grande para evitar pérdidas de paquetes y que los paquetes que llegan al primer nodo conectado a la LAN sean consecutivos pero conserven su comportamiento a ráfagas propio de la fuente de vídeo. Las medidas de delay y jitter se toman entre los nodos N1 y N2 que sí están conectados por un tramo de LAN con EEE.

CAPÍTULO 3. PRUEBAS REALIZADAS

En este capítulo presentamos los resultados de las simulaciones realizadas y los analizamos para extraer las conclusiones de cada prueba. Se exponen todos los escenarios simulados pero de todas las medidas que se han tomado, pero sólo se analizarán las que resultan significativas relegando a los anexos el resto. Como se indica en los capítulos anteriores las simulaciones de todos los escenarios están basadas en el estándar 1000BASE-T.

3.1. TCP

El modelo de TCP que hemos tomado como referencia es el TCP Linux, principalmente por coherencia con trabajos anteriores realizados con este mismo modelo de TCP de cara a la comparación de resultados. Se ha tomado como premisa realizar las simulaciones con un valor máximo para la ventana de transmisión de TCP suficientemente grande para que no suponga una limitación a la consecución de la ventana óptima durante la transmisión para ninguno de los round-trip-times (RTT) de las simulaciones. Todos los escenarios en los que no se indique lo contrario están configurados con TCP-Linux y una ventana de transmisión óptima.

3.1.1. Un enlace TCP

Como se indica previamente a la explicación de la elección de escenarios simulados aunque tomamos como referencia TCP Linux hemos realizado adicionalmente simulaciones para el escenario básico de TCP (un enlace con tráfico FTP) con las variantes Reno y Vegas. A continuación se exponen los resultados de las diferentes variantes.

3.1.1.1. TCP Linux

Fig. 3.1 Escenario de la simulación

<u>Características de la simulación</u>

Protocolo: TCP-Linux
Fuente de datos: FTP, continua.
Tamaño trama: 1526 bytes, payload: 1500 bytes
Volumen de datos enviado: 10 Mbytes

Parámetro que varía: retardo del enlace, en el rango 1-5000 µs.

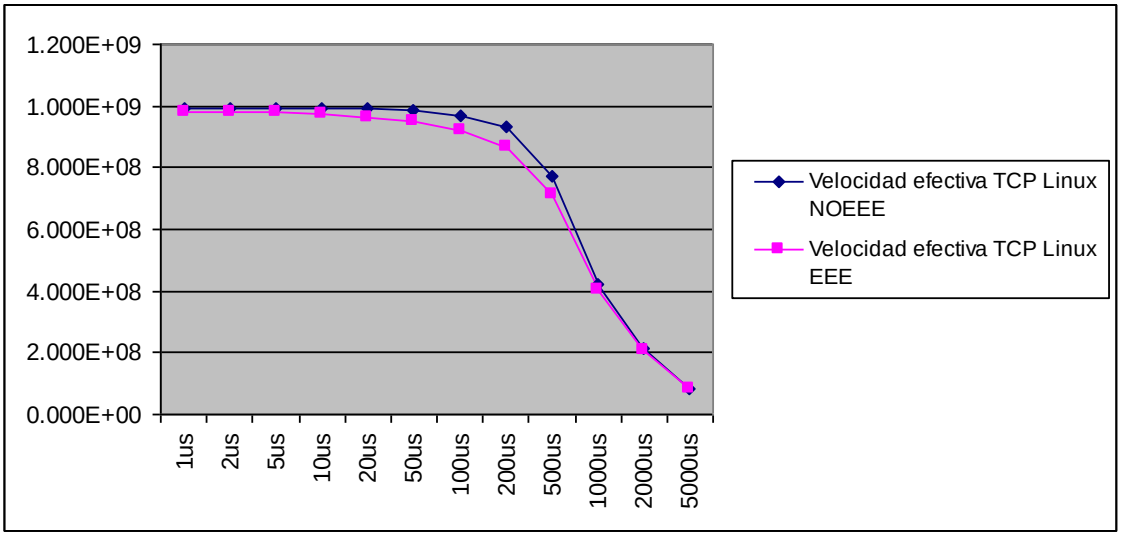

Fig. 3.2 Velocidad efectiva en función del delay de enlace

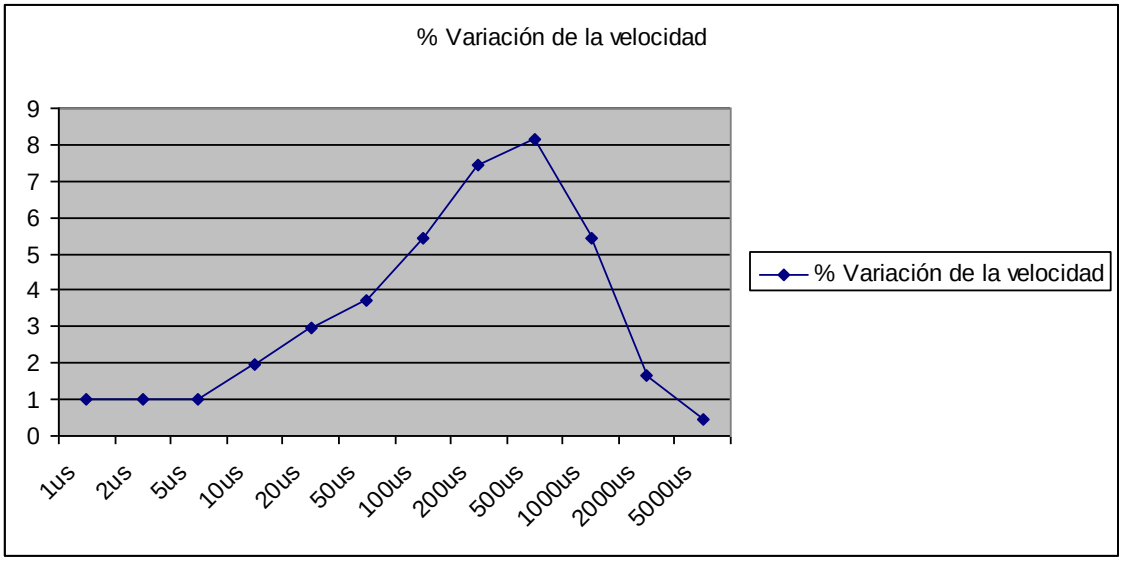

Fig. 3.3 Variación de la velocidad efectiva NOEEE/EEE

Para este escenario en que la tarjeta Ethernet estará transmitiendo durante casi todo el tiempo se espera que la diferencia de velocidad aplicando EEE sea mínima. Se observa que nuestras expectativas se cumplen y aunque la velocidad efectiva de la simulación en que se aplica EEE es ligeramente inferior a la obtenida sobre Ethernet convencional, esta variación se mantiene por debajo del 10%.

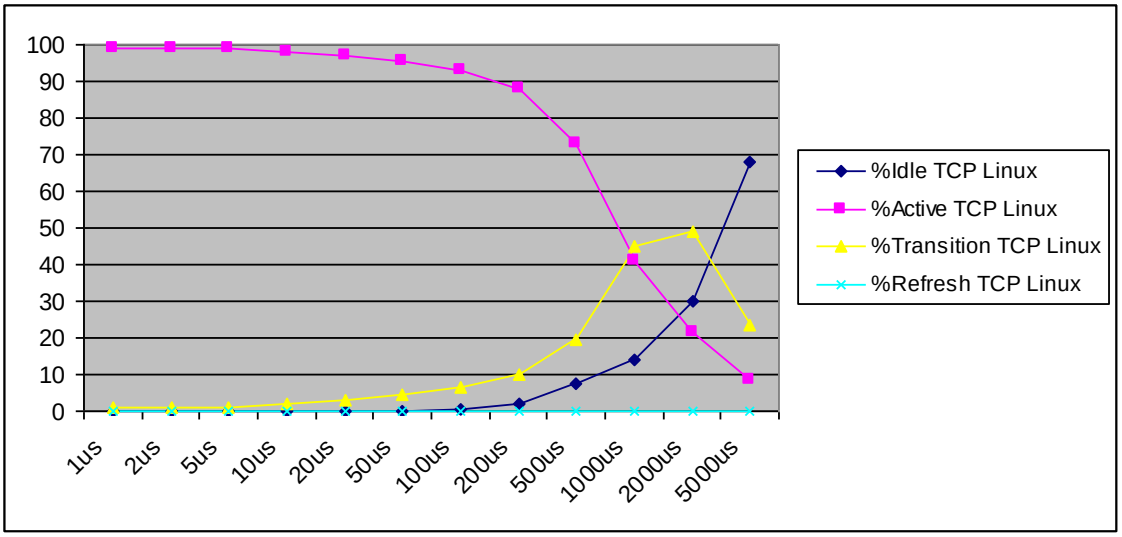

Fig. 3.4 Porcentaje de tiempo en cada estado en función del delay de enlace

Como contrapartida una utilización alta del enlace nos limita el tiempo en que la tarjeta Ethernet se encuentra en estado de Idle y por tanto el ahorro potencial en este caso será mínimo, pues hasta que no llegamos a un delay de enlace de 5000 µs no superamos el 50% de tiempo en el estado de Idle.

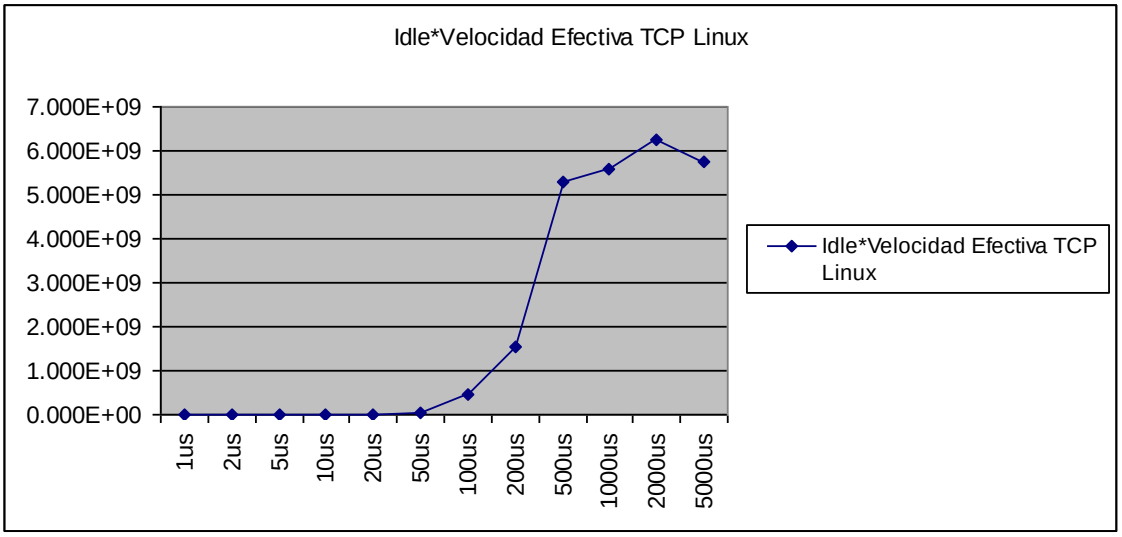

Fig. 3.5 Producto Idle * Velocidad efectiva en función del delay de enlace

El producto Idle por velocidad efectiva nos muestra una zona de compromiso entre la velocidad efectiva de la transmisión y el ahorro potencial. Encontramos una zona óptima entre los 500 µs y los 5000 µs, debido a que para delays de enlace mas bajos el tiempo que la tarjeta Ethernet pasa en idle resulta mínimo.

3.1.1.2. TCP Reno

Fig. 3.6 Escenario de la simulación

Características de la simulación

Protocolo: TCP-Reno
Fuente de datos: FTP, continua.
Tamaño trama: 1526 bytes, payload: 1500 bytes
Volumen de datos enviado: 10 Mbytes
Parámetro que varía: retardo del enlace, en el rango 1-5000 µs.

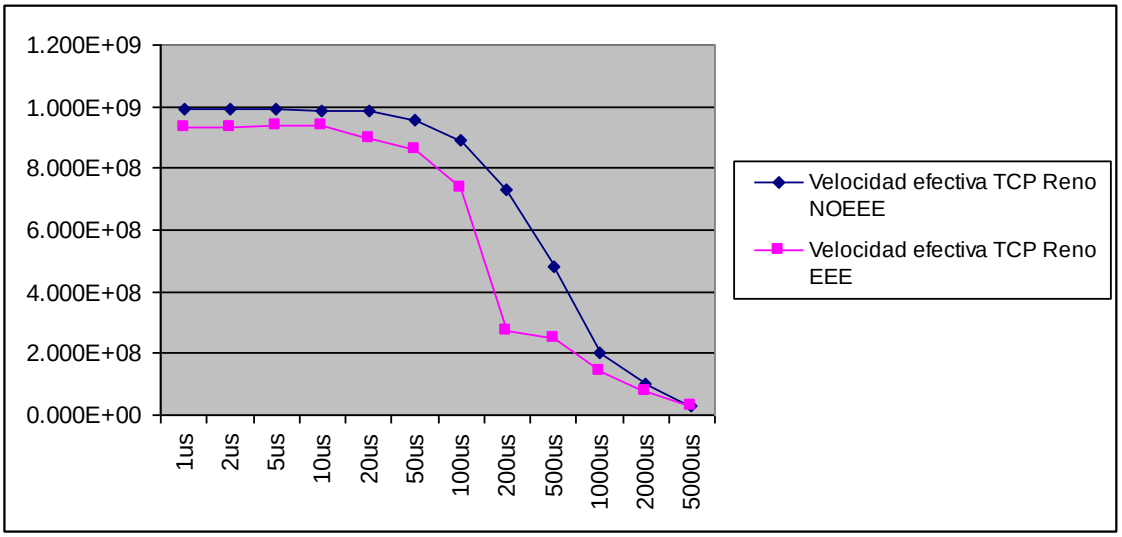

Fig. 3.7 Velocidad efectiva en función del delay de enlace

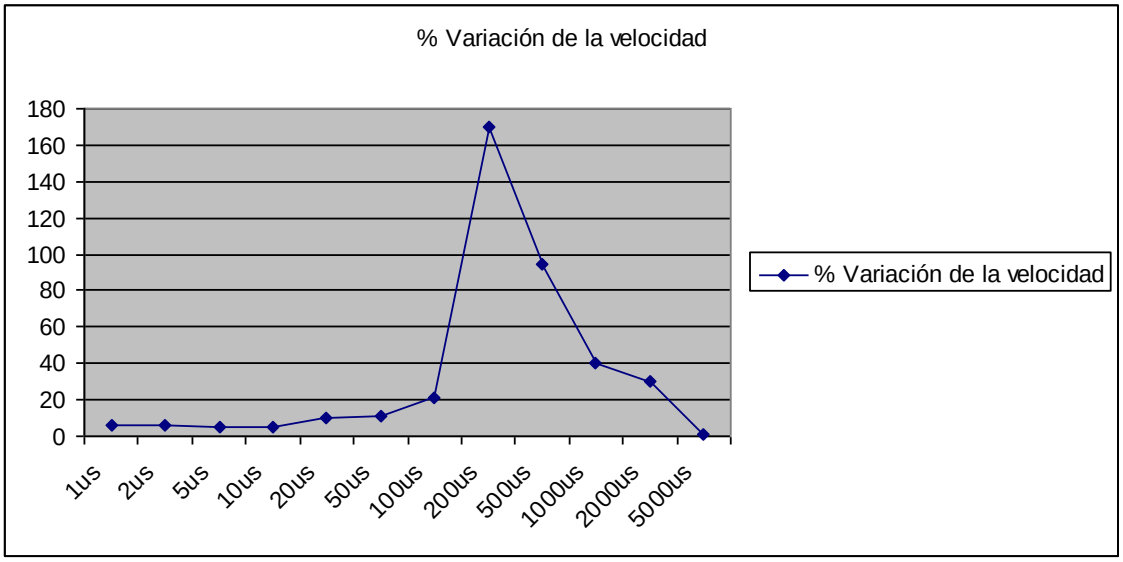

Fig. 3.8 Variación de la velocidad efectiva NOEEE/EEE

Para la variante TCP Reno observamos una tendencia general similar a la de la variante TCP Linux, lo cual también encaja con nuestras expectativas debido a que tanto TCP Reno como TCP Linux implementa un algoritmo de control de congestión basado en las pérdidas. Sin embargo para esta variante la disminución de la velocidad efectiva es ligeramente mayor, variación que alcanza su máximo en el escenario para un valor de enlace de 200 µs.

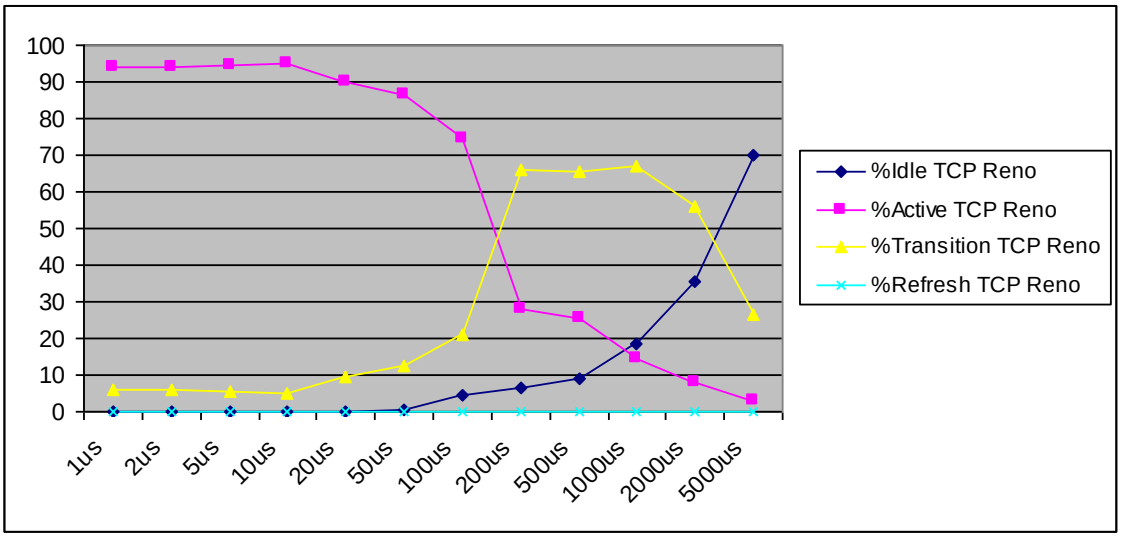

Fig. 3.9 Porcentaje de tiempo en cada estado en función del delay de enlace

Nuestra interpretación de este efecto es que se debe a que el tiempo de sleep para 1000BASE-T es de 182 µs y a causa de ello el ritmo de crecimiento de la ventana en la implementación para ns-2 de esta variante de TCP se ve más afectado por el ciclo Wake-Sleep al llegar a un delay de enlace de 200 µs, y se produce un abrupto descenso de la velocidad efectiva. Aunque no hemos

llegado a determinar la dinámica de la causa de la caída de velocidad efectiva en este punto, descartamos las pérdidas de paquetes como causa de este efecto ya que no se producen y la configuración del escenario debido que es el mas sencillo posible. Este efecto provoca que a pesar de la disminución del porcentaje de tiempo que la tarjeta pasa en active, el porcentaje de tiempo que pasa en idle no aumente pero si lo haga el de transición entre estados que se dispara en este punto y ocasiona que disminuya la velocidad sin obtener beneficio adicional alguno en el ahorro de energía.

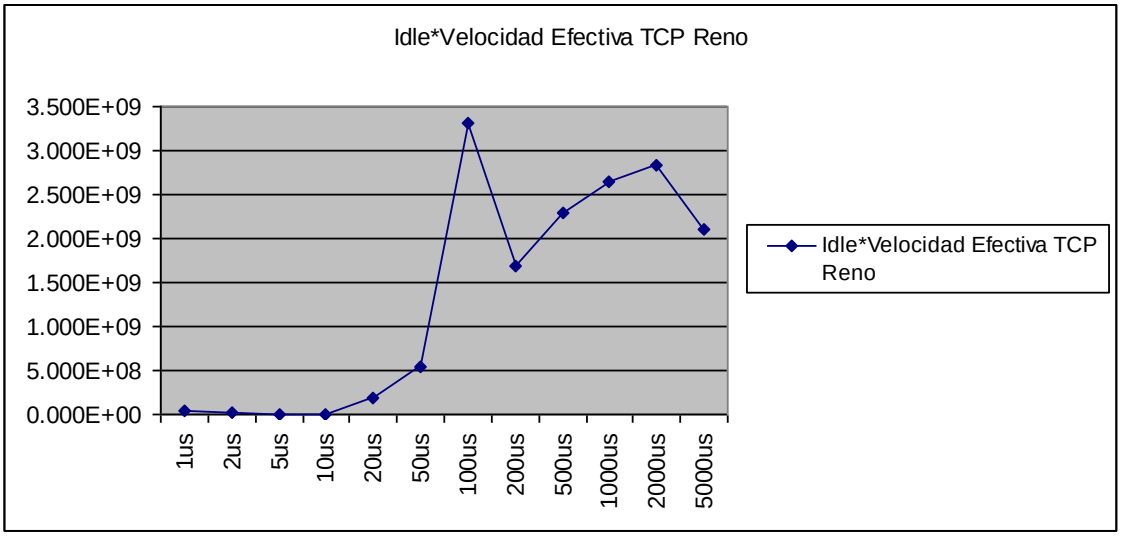

Fig. 3.10 Producto Idle * Velocidad efectiva en función del delay de enlace

Para esta variante de TCP vemos que en general el producto Idle por velocidad efectiva es algo más bajo que para TCP Linux pero mantiene la misma tendencia con la diferencia del repentino aumento de transiciones de estados y la consecuente disminución de velocidad efectiva a partir de 200 µs, la cual al reducir la velocidad efectiva produce también una disminución del producto idle por velocidad efectiva. Aunque posteriormente continúa de nuevo la tendencia al alza a partir de este punto, no llega a alcanzar valores tan elevados como en la variante anterior.

3.1.1.3. TCP Vegas

Fig. 3.11 Escenario de la simulación

<u>Características de la simulación</u>

Protocolo: TCP-Vegas
Fuente de datos: FTP, continua.
Tamaño trama: 1526 bytes, payload: 1500 bytes
Volumen de datos enviado: 10 Mbytes
Parámetro que varía: retardo del enlace, en el rango 1-5000 µs.

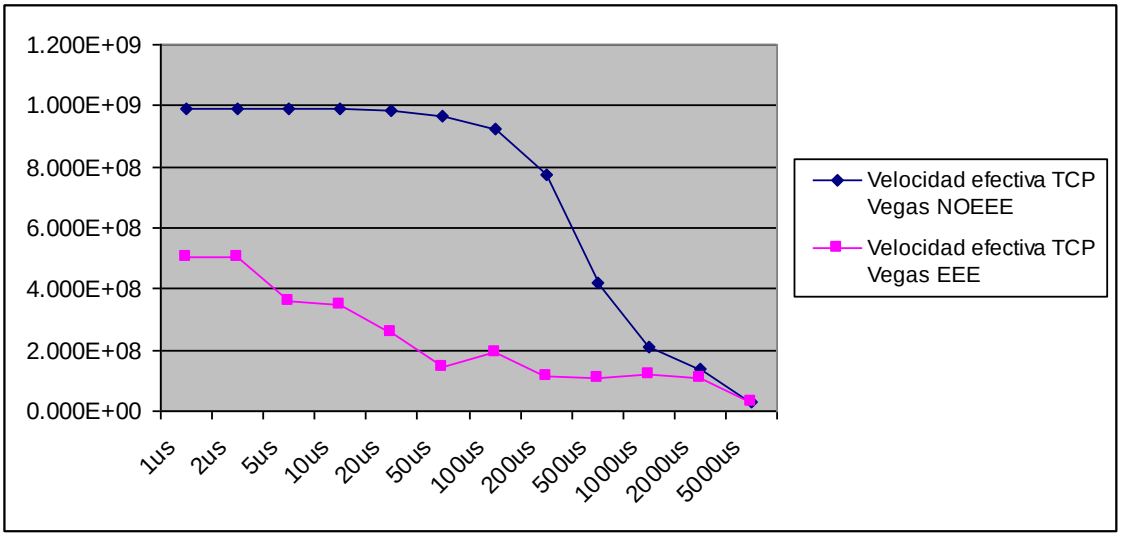

Fig. 3.12 Velocidad efectiva en función del delay de enlace

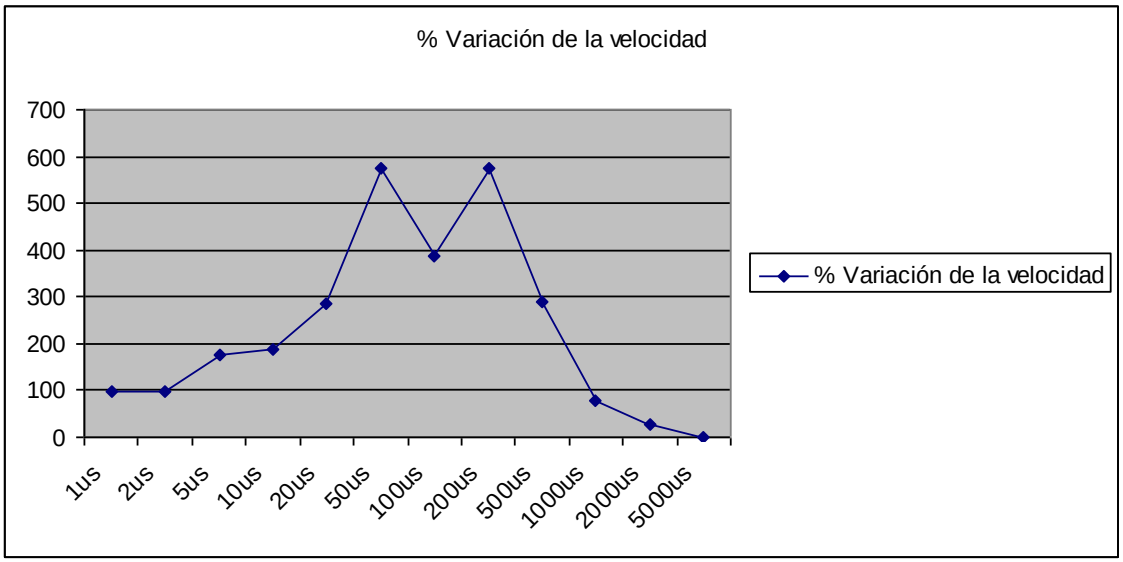

Fig. 3.13 Variación de la velocidad efectiva NOEEE/EEE

Tal como se había previsto a diferencia de en las dos versiones anteriores, en TCP-Vegas la variación de la velocidad efectiva de EEE con respecto a Ethernet convencional es mucho mayor, siendo la velocidad conseguida con EEE inferior o cercana a la mitad de la lograda con Ethernet convencional para delays de enlace de hasta 1000 µs. Esto es debido al control de congestión que realiza TCP-Vegas que se basa no solo en la pérdida de paquetes sino

también en la medida del RTT. Los tiempos de transición de estados de EEE suponen una variación importante para el RTT, a los que TCP-Vegas reacciona cerrando la ventana de transmisión. Esto solamente produce que la tarjeta Ethernet entre en estado de Sleep en más ocasiones causando de nuevo que la variación entre RTTs sea mayor y provocando una notable disminución de la velocidad efectiva.

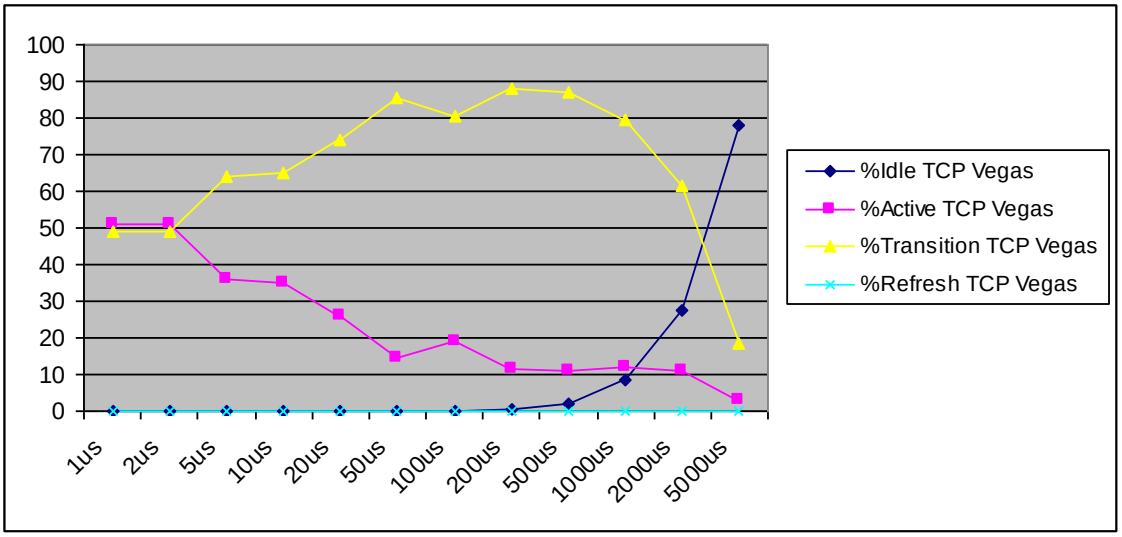

Fig. 3.14 Porcentaje de tiempo en cada estado en función del delay de enlace

Como consecuencia de esta dinámica el tiempo de transición entre estados es el predominante y a pesar de la caída de la velocidad efectiva no obtenemos porcentajes significativos de tiempo de Idle y por tanto ahorros de energía importantes hasta delays de enlace de 2000 µs.

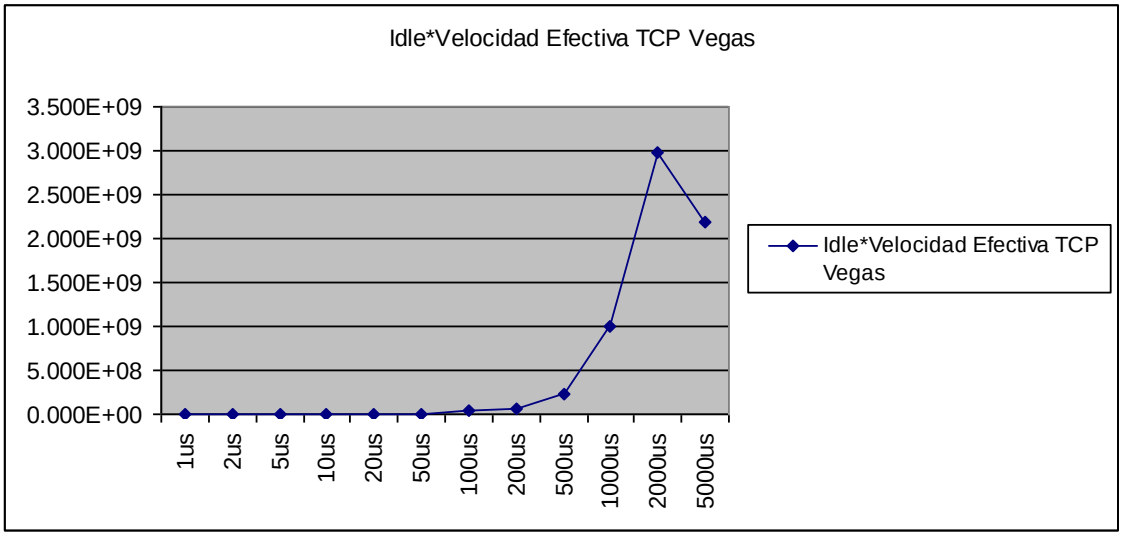

Fig. 3.15 Producto Idle * Velocidad efectiva en función del delay de enlace

La zona óptima en que el compromiso entre ahorro de energía y velocidad efectiva también se estrecha de forma drástica quedando reducida prácticamente al escenario con delay de enlace de 2000 µs.

3.1.2. Dos enlaces TCP

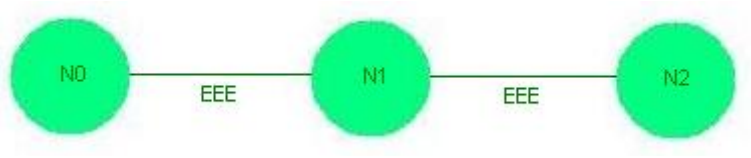

Fig. 3.16 Escenario de la simulación

Características de la simulación

Protocolo: TCP-Linux
Fuente de datos: FTP, continua.
Tamaño trama: 1526 bytes, payload: 1500 bytes
Volumen de datos enviado: 10 Mbytes
Parámetro que varía: retardo del enlace, en el rango 1-5000 µs.

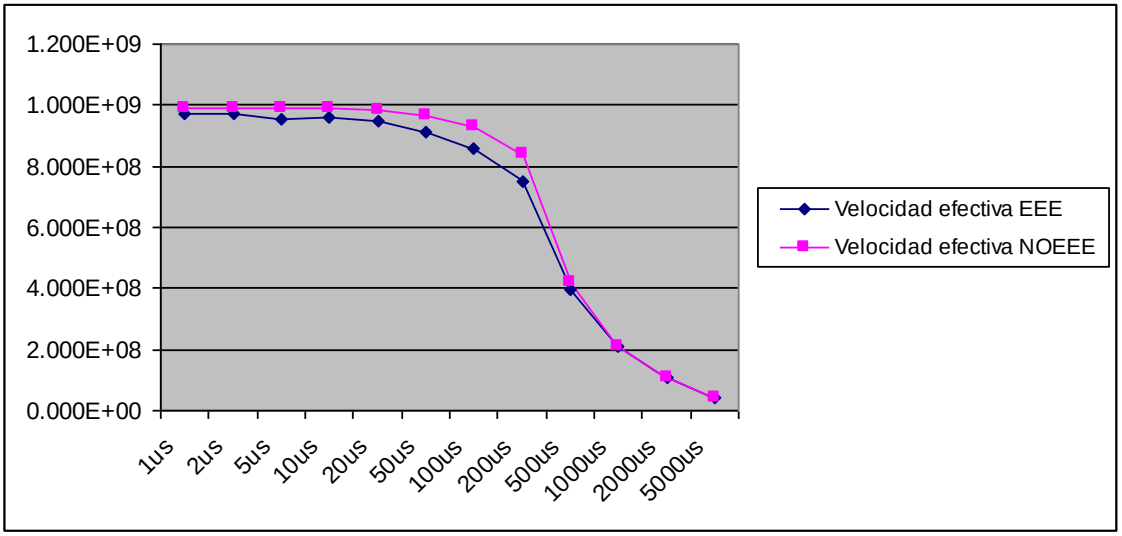

Fig. 3.17 Velocidad efectiva en función del delay de enlace

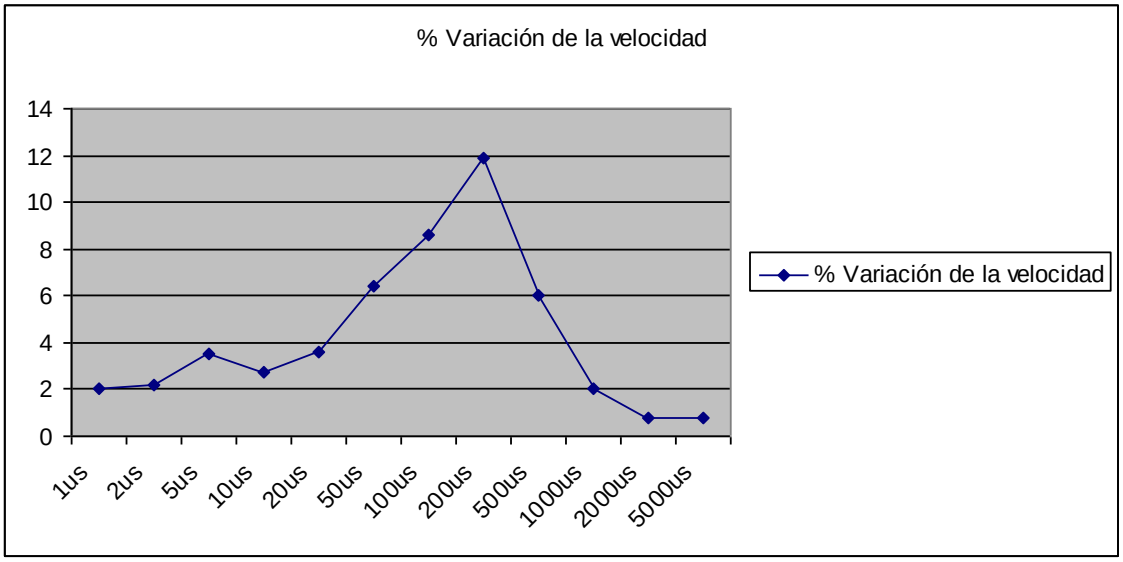

Fig. 3.18 Variación de la velocidad efectiva NOEEE/EEE

De nuevo para dos enlaces EEE obtenemos una velocidad efectiva ligeramente inferior a la medida sobre Ethernet clásico. Aunque la disminución de la velocidad es ligeramente mayor que para un solo enlace la diferencia es muy poco significativa, conservando la misma tendencia y alcanzando su punto máximo en el escenario de 200 µs donde la velocidad efectiva es un 12% inferior.

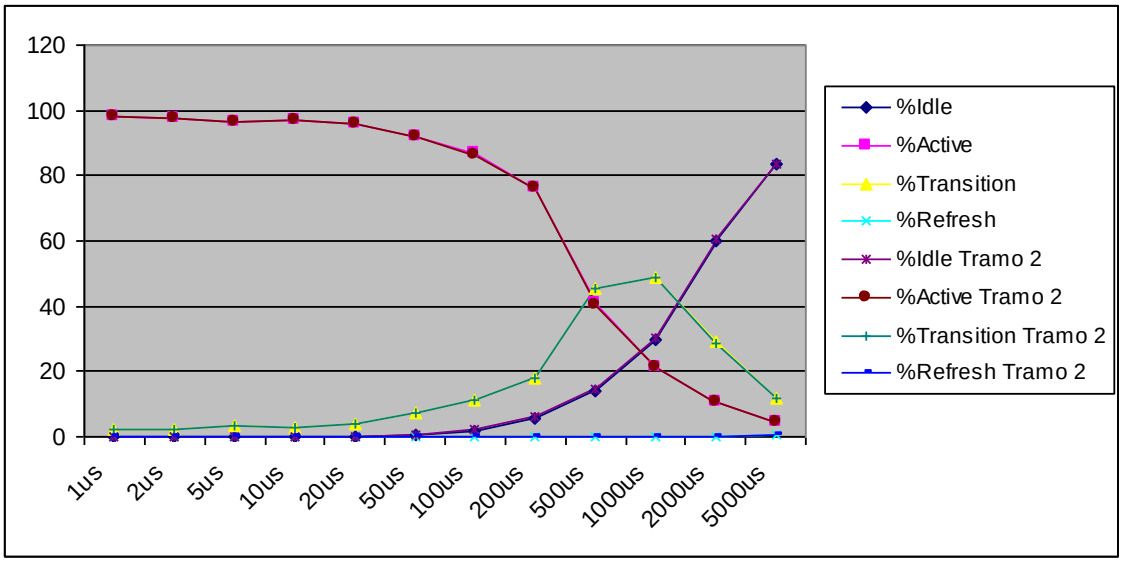

Fig. 3.19 Porcentaje de tiempo en cada estado en función del delay de enlace

Lo mismo ocurre con respecto al porcentaje de tiempo que la tarjeta Ethernet pasa en cada estado, sigue la misma tendencia que para el escenario con un solo tramo de LAN EEE. Se observa que el porcentaje de tiempo que la tarjeta Ethernet pasa en cada estado es el mismo para el primer y segundo tramo.

Esto ocurre en todos los casos simulados por lo que en los gráficos posteriores solamente se presentará el porcentaje de tiempo que la primera tarjeta Ethernet pasa en cada uno de los estados, relegando el resto de gráficos a los anexos.

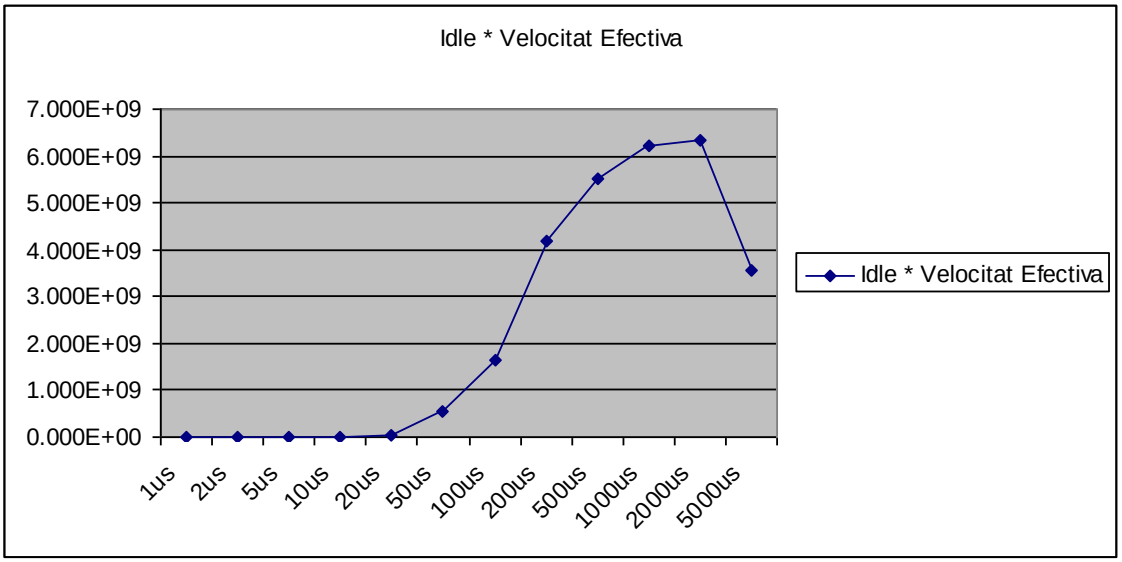

Fig. 3.20 Producto Idle * Velocidad efectiva en función del delay de enlace

Siguiendo la misma tendencia que para un solo tramo EEE encontramos que en este caso el aumento del producto Idle por velocidad efectiva presenta una curva más suave y esta desplazado hacia la izquierda. Presentando la zona óptima entre 500 µs y 2000 µs. Este desplazamiento es consecuencia del aumento del RTT que implica un enlace más. Se observa que para dos enlaces EEE consecutivos el efecto de EEE sobre la dinámica TCP sigue la misma tendencia que para un solo enlace siendo la diferencia entre ambos escenarios mínima.

3.1.3. Tres enlaces TCP

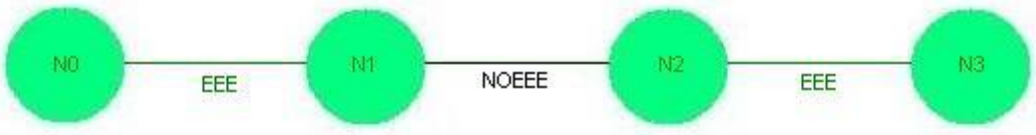

Fig. 3.21 Escenario A de la simulación

Fig. 3.22 Escenario B de la simulación

Características de la simulación

Protocolo: TCP-Linux
Fuente de datos: FTP, continua.
Tamaño trama: 1526 bytes, payload: 1500 bytes
Volumen de datos enviado: 10 Mbytes
Parámetro que varía: retardo del enlace, en el rango 1-5000 µs.

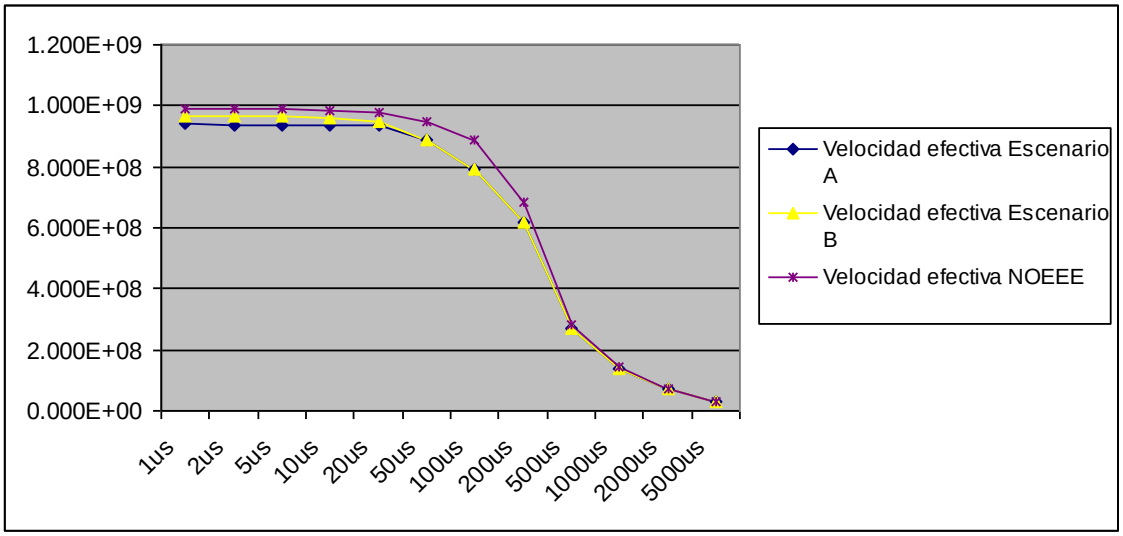

Fig. 3.23 Velocidad efectiva en función del delay del enlace

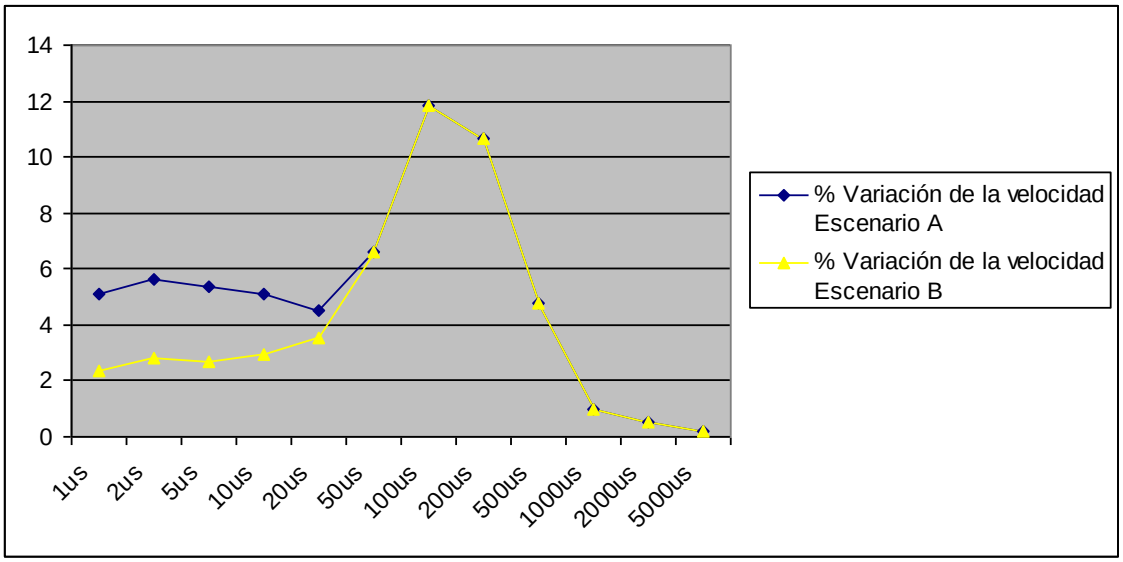

Fig. 3.24 Variación de la velocidad efectiva NOEEE/EEE

Como esperábamos, la velocidad efectiva de los escenarios que implementan EEE y Ethernet clásico siempre es inferior a la de los que únicamente implementan Ethernet clásico. Sin embargo en el escenario A en el cual los

tramos EEE no son consecutivos la diferencia de velocidad efectiva con respecto a Ethernet clásico es mayor para las simulaciones con menor delay de enlace. Esto implica que el RTT medio es más elevado para el escenario A que para el escenario B a causa de que se producen más transiciones entre estados. Como en los escenarios anteriores se descartó la perdida de paquetes como causa de este efecto ya que no se producen. La variación del delay de extremo a extremo en ambos escenarios también resulta mínima para la transmisión de los paquetes de datos y a pesar de que no se logró discernir la dinámica concreta de este efecto si se determinó que era una variación relativa a la transmisión de los ACK.

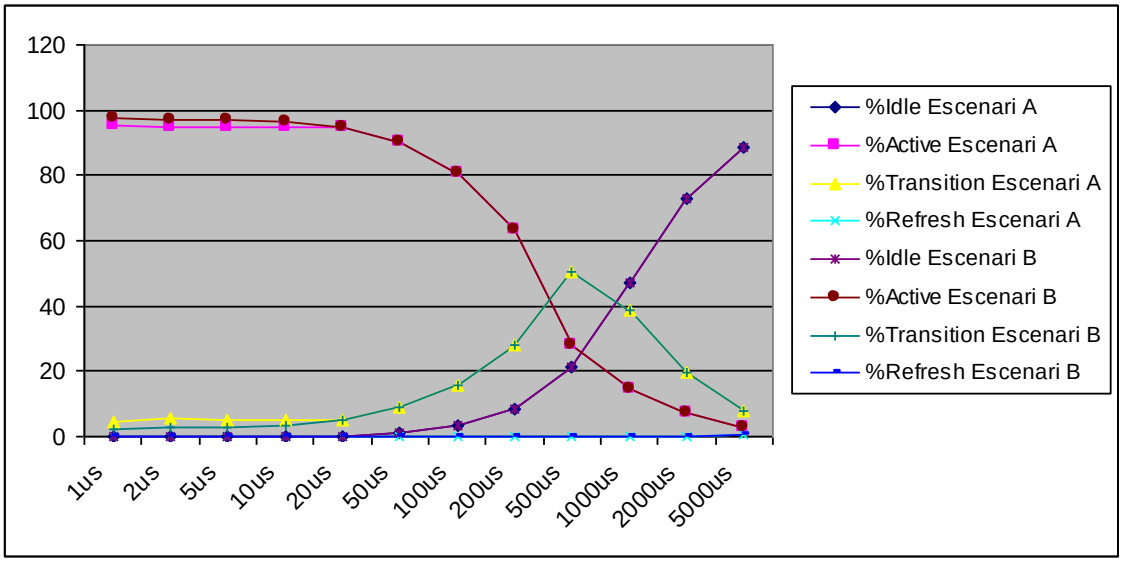

Fig. 3.25 Porcentaje de tiempo en cada estado en función del delay del enlace

Al analizar el porcentaje de tiempo que las tarjetas Ethernet pasan en cada estado se observa que para la franja con menor delay de enlace el porcentaje de tiempo de active que se pierde, lo hace a favor del porcentaje de tiempo de transición entre estados. Por tanto esta es una pérdida de velocidad que no implica ahorro de energía.

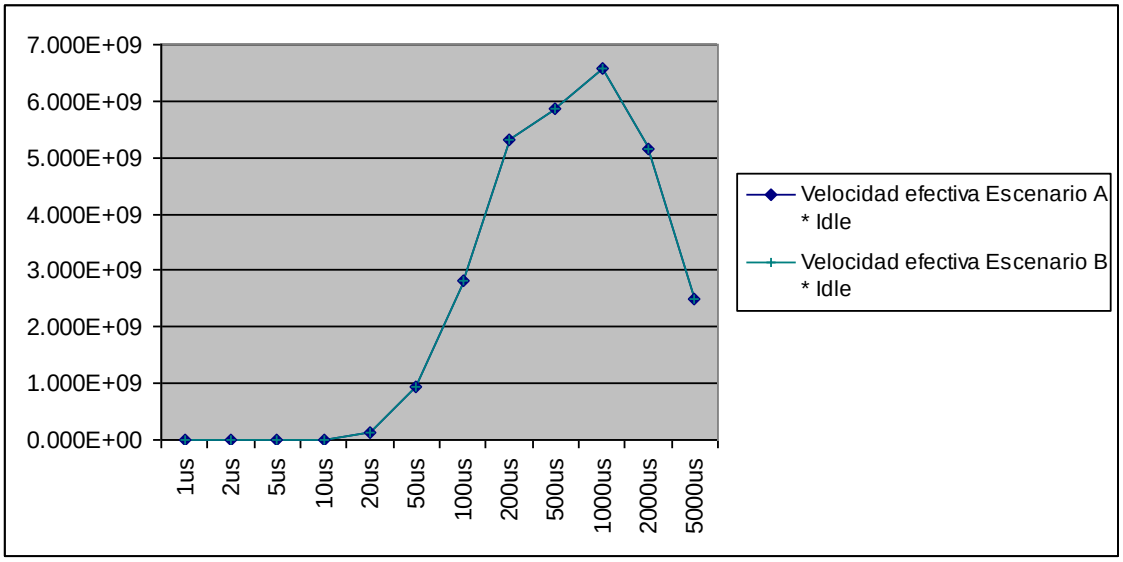

Fig. 3.26 Producto Idle * Velocidad efectiva en función del delay del enlace

El producto Idle por velocidad efectiva en cambio no se ve afectado por la diferencia entre ambos escenarios, ya que la zona óptima se sitúa en delays de enlace más elevados que ya no se ven alterados por este efecto. Se observa que sigue la misma tendencia, desplazándose hacia la izquierda con respecto al escenario de un solo enlace y alcanzando una franja óptima para delays de enlace de entre 200 µs y 2000 µs.

3.2. TCP con ventana limitada

3.2.1. Un enlace TCP

Fig. 3.27 Escenario de la simulación

<u>Características de la simulación</u>

Protocolo: TCP-Linux
Fuente de datos: FTP, continua.
Tamaño trama: 1526 bytes, payload: 1500 bytes
Volumen de datos enviado: 10 Mbytes
Parámetro que varía: retardo del enlace, en el rango 1-5000 µs.
Tamaño máximo de la ventana de transmisión: 30000 bytes

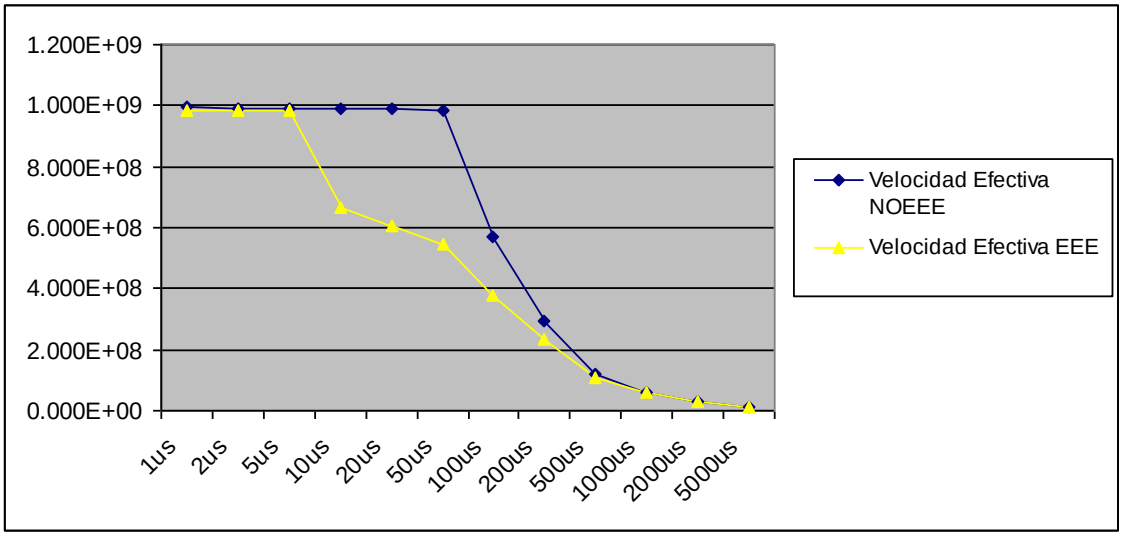

Fig. 3.28 Velocidad efectiva en función del delay del enlace

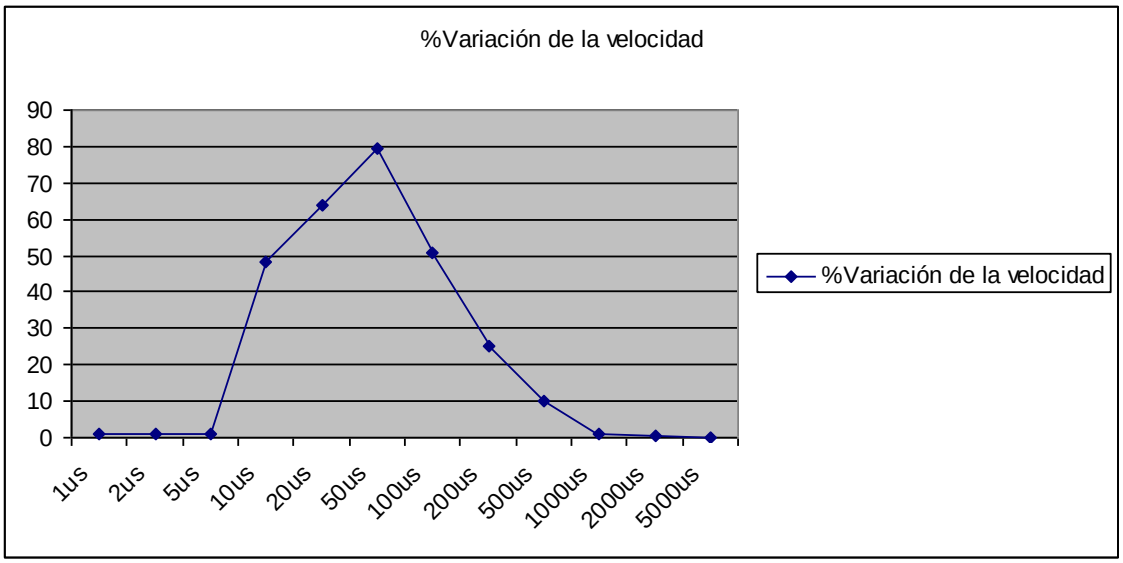

Fig. 3.29 Variación de la velocidad efectiva NOEEE/EEE

Se observa que con una ventana de congestión limitada el efecto de los mecanismos de ahorro de energía de EEE sobre la velocidad efectiva es mucho mayor en los escenarios con un delay de enlace superior a 5 µs, llegando a haber una diferencia de velocidad de hasta el 80%. Esto es debido a que el delay adicional introducido por los mecanismos de ahorro de energía de EEE aumenta el RTT hasta el punto de que deja de cumplirse la condición de ventana óptima a partir de los 10 µs, donde se ve una abrupta disminución de la velocidad efectiva. En el caso de Ethernet clásico esta condición deja de cumplirse a partir de un delay de enlace de 100 µs, donde se observa una disminución similar al punto anteriormente mencionado para EEE.

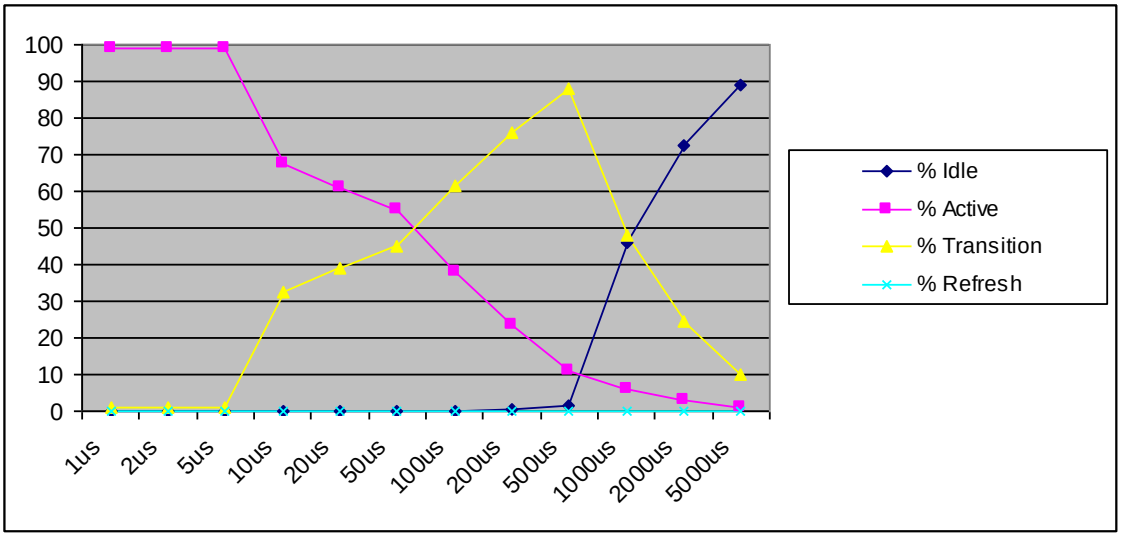

Fig. 3.30 Porcentaje de tiempo en cada estado en función del delay del enlace

Esta disminución de velocidad implica una disminución de tiempo en que la tarjeta Ethernet se encuentra en estado Active. Sin embargo, el tiempo en que la tarjeta se encuentra en los estados de transición entre Active e Idle crece de forma proporcional a la disminución del tiempo de Active. Por tanto no empezamos a tener un porcentaje elevado de tiempo de Idle hasta los 1000 µs de delay de enlace.

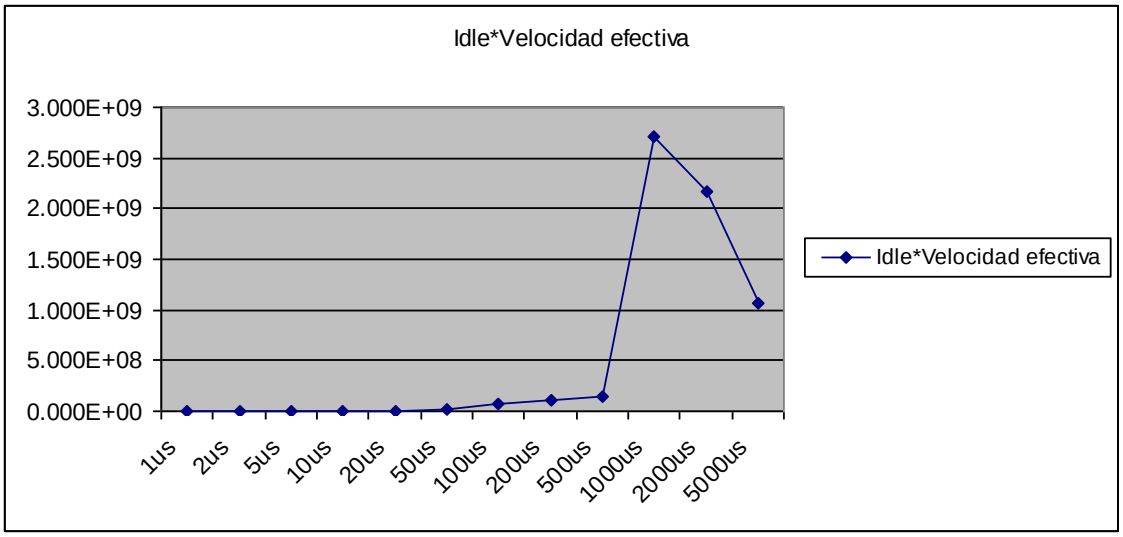

Fig. 3.31 Producto Idle * Velocidad efectiva en función del delay del enlace

A pesar de que el producto Idle por velocidad efectiva también es menor, debido a que la velocidad efectiva es menor como consecuencia de la limitación de la ventana de transmisión, se observa que sigue la misma tendencia que para una ventana de transmisión óptima.

3.3. UDP

3.3.1. Un enlace UDP

Fig. 3.32 Escenario de la simulación

Características de la simulación

Protocolo: UDP
Tipos de datos: CBR – tasa constante y continua
Tamaño trama: 1526 bytes
Payload: 1500 bytes

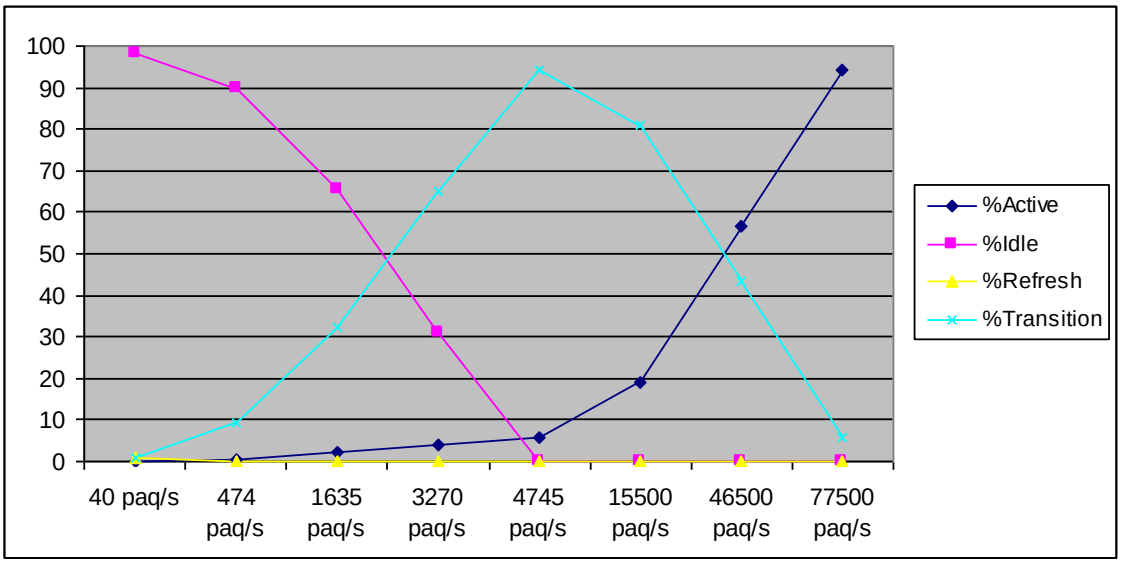

Fig. 3.33 Porcentaje de tiempo en cada estado en función de la velocidad

Al carecer el protocolo UDP de mecanismos de control de congestión, el efecto de EEE es mucho más simple. Para velocidades de transferencia de datos más bajas la utilización del enlace es inferior, lo que implica que el tiempo en que la tarjeta Ethernet se encuentra en estado Idle es mayor y por tanto se consigue un mayor ahorro de energía. A medida que se aumenta la velocidad de transmisión de datos el tiempo que la tarjeta pasa en estado Idle disminuye y el tiempo de transición entre estados aumenta proporcionalmente hasta un punto crítico para la tasa de 4745 paquetes por segundo.

El cálculo de este punto crítico se ha llevado a cabo a partir de los siguientes parámetros:

Tabla 3.1. Parámetros para el cálculo del punto crítico

Parámetro	Tiempo
Twake	16,5 µs
Tsleep	182 µs
Ttx	12,208 µs

$$Limite = \frac{1}{Twake + Tsleep + Ttx} = \frac{1}{210{,}708\mu s} = 4745\, paquetes\ /\ segundo \qquad \textbf{(3.1)}$$

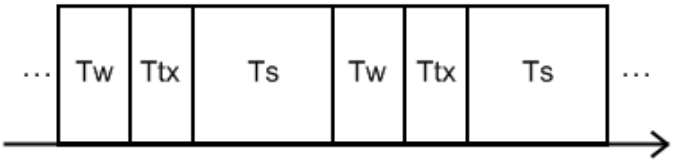

Fig. 3.34 Situación límite en la tasa de 4745 paquetes por segundo

En este punto la tarjeta nunca entra en el estado Idle porque cada vez que completa el ciclo Wake-Active-Sleep llega un paquete nuevo y de nuevo entra en el estado Wake. A partir de esta tasa el tiempo que la tarjeta pasa en estado LPI es 0 y por tanto no se consigue ahorro de energía.

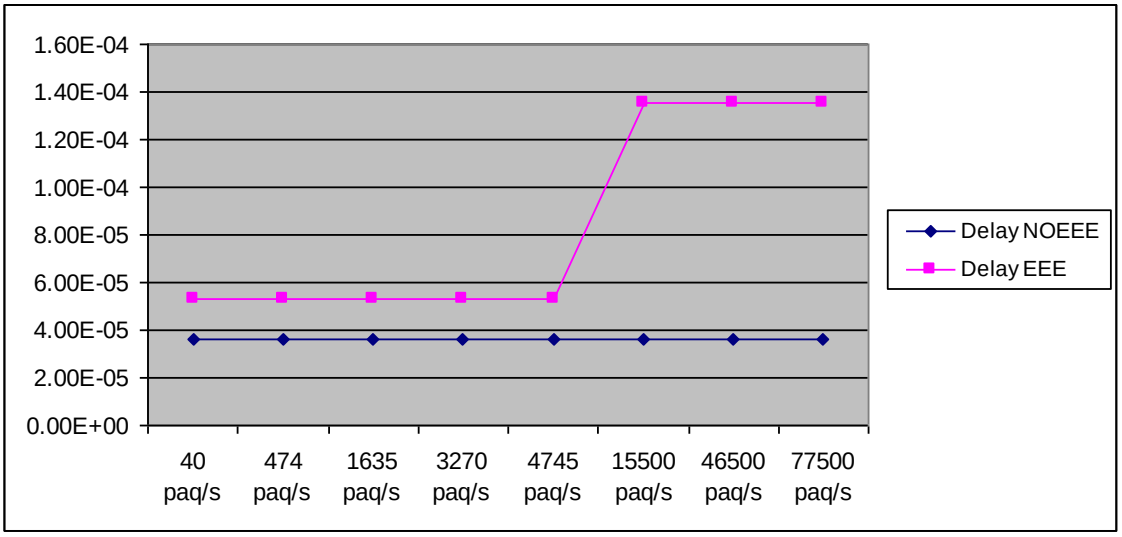

Fig. 3.35 Delay en función de la velocidad

Se observa que el delay de extremo a extremo con respecto a la implementación clásica de Ethernet se mantiene estable hasta el punto crítico en 4745 paquetes por segundo, un tiempo de Wake por encima de este. Y a partir de ese punto experimenta una subida cuando el tiempo de Sleep provoca un delay adicional, esto se debe a que cuando se inicia una transición de estado no se pasa a la siguiente hasta que la primera se ha completado.

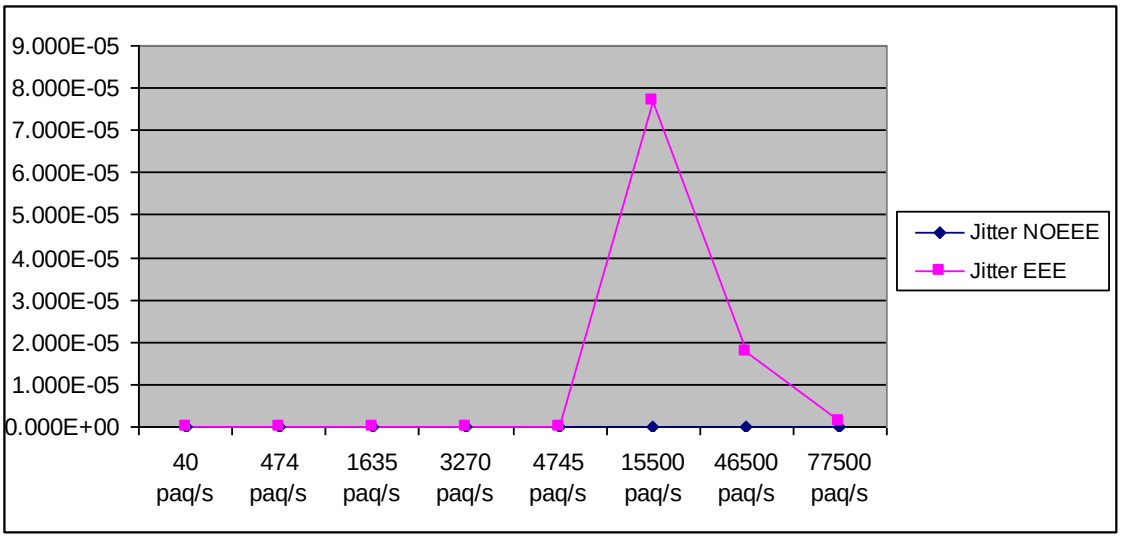

Fig. 3.36 Jitter en función de la velocidad

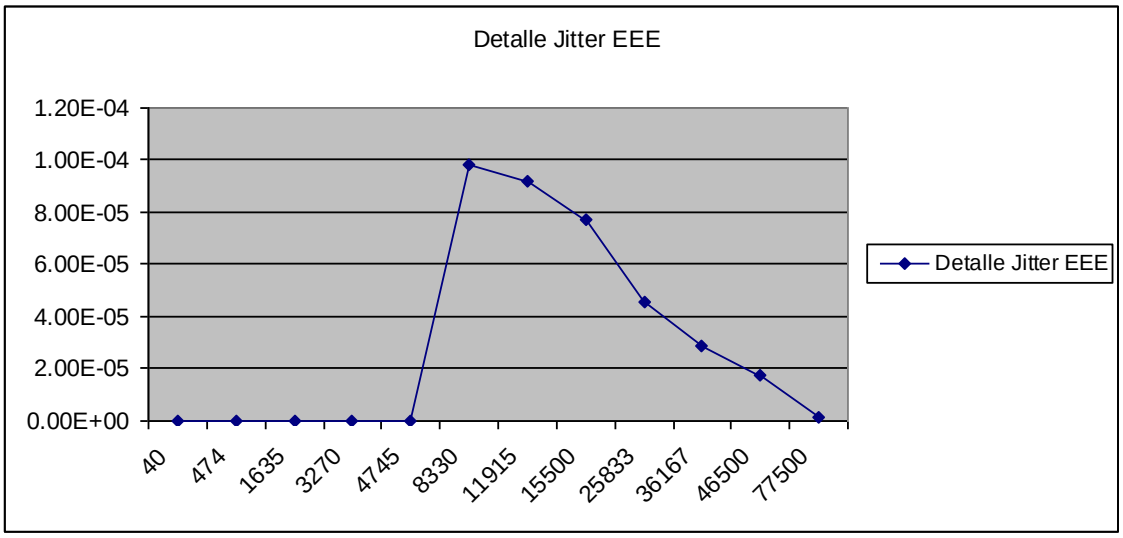

Fig. 3.37 Detalle del jitter en función de la velocidad

El comportamiento del jitter también es estable hasta el punto crítico de 4745 paquetes por segundo, tomando un valor de de 0 hasta ese punto. A partir de ahí se dispara tomando el valor máximo en la situación que se transmiten dos paquetes por cada ciclo Wake-Active-Sleep de forma que la diferencia entre el delay de un paquete y el anterior siempre es muy elevada. Al aumentar más la tasa va llegando cada vez un número mayor de paquetes en el tiempo que dura el ciclo y el jitter (diferencia de delay entre paquetes consecutivos) va disminuyendo progresivamente.

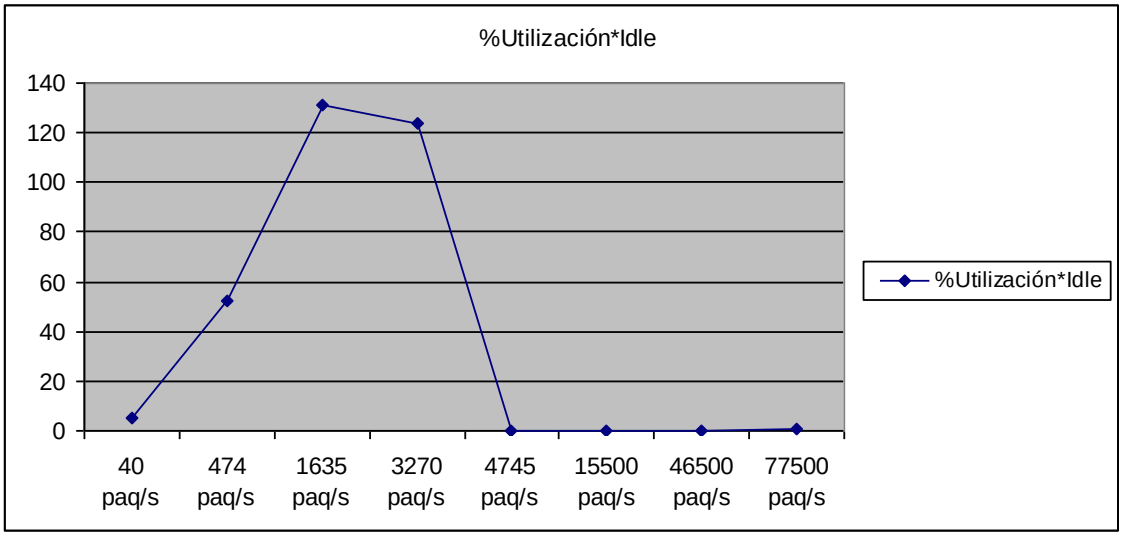

Fig. 3.38 Producto Idle * Utilización

El producto utilización por idle únicamente tiene un valor elevado para tasas por debajo de 4745 paquetes por segundo ya que a partir de ese momento deja de entrarse en estado de LPI. El gráfico 3.38 muestra visualmente que como solamente se obtiene un ahorro de energía significativo para utilizaciones bajas del enlace la franja óptima entre velocidad de transmisión y ahorro también es cercana a esas situaciones de baja utilización del enlace.

3.3.2. Dos enlaces UDP

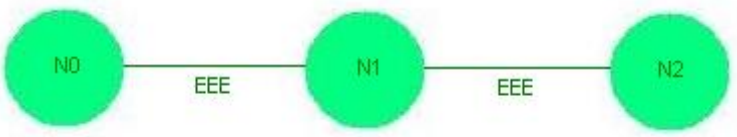

Fig. 3.39 Escenario de la simulación

<u>Características de la simulación</u>

Protocolo: UDP
Tipos de datos: CBR – tasa constante y continua
Tamaño trama: 1526 bytes
Payload: 1500 bytes

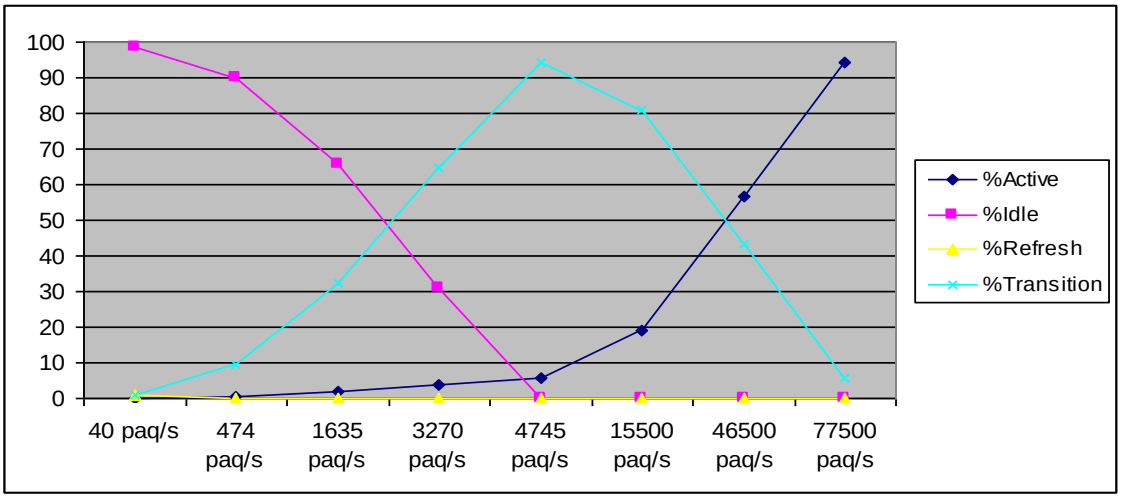

Fig. 3.40 Porcentaje de tiempo en cada estado en función de la velocidad

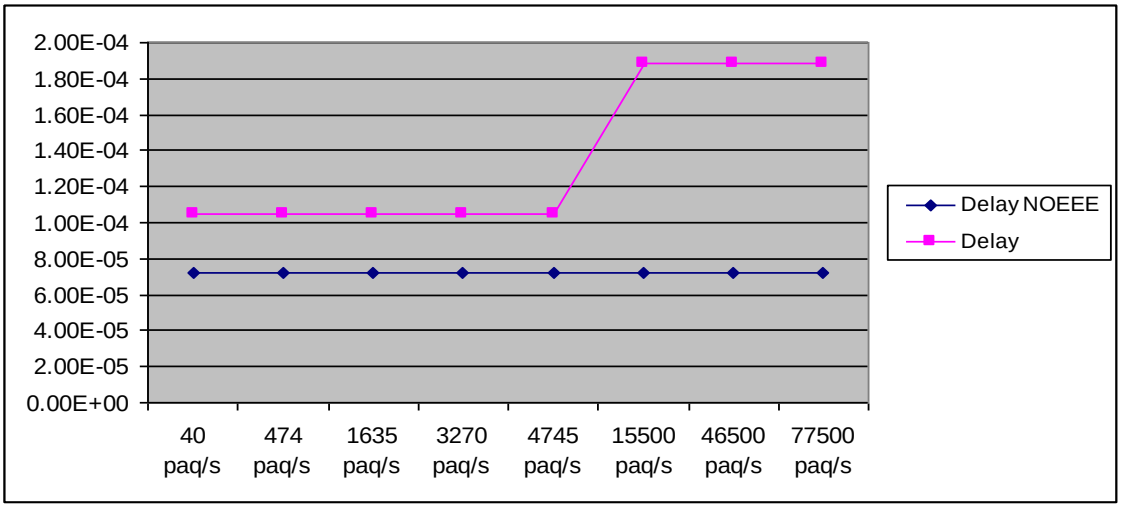

Fig. 3.41 Delay en función de la velocidad

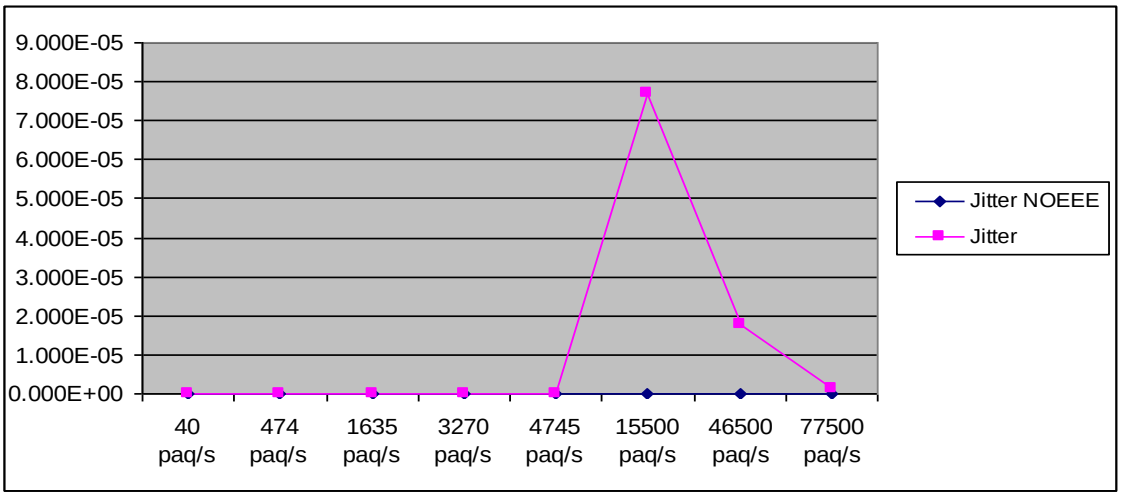

Fig. 3.42 Jitter en función de la velocidad

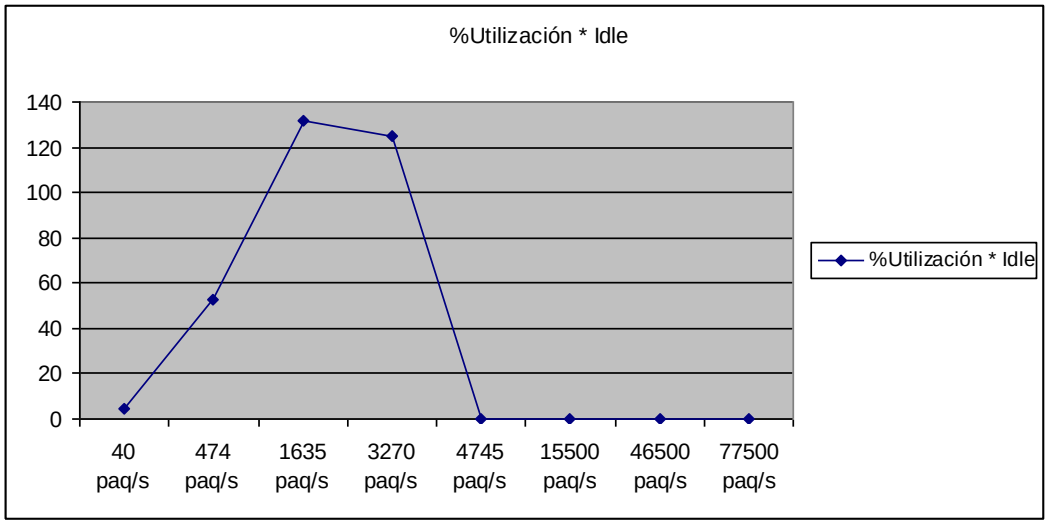

Fig. 3.43 Producto Idle * Utilización

Para dos enlaces los delays de extremo a extremo son mayores, pero como UDP no tiene ninguna clase de mecanismo de confirmación que controle las pérdidas ni control de congestión que mida los delays ni RTTs, el comportamiento que obtenemos es el mismo que para un solo enlace. Encontramos los mismos puntos críticos y zonas de potencial ahorro de energía que para un solo enlace.

3.3.3. Tres enlaces UDP

En las simulaciones del protocolo UDP sobre los escenarios A y B no se observa ninguna tendencia diferente a las que ya se han visto previamente en los escenarios de uno y dos enlaces. Los gráficos pertenecientes a estas simulaciones pasan a formar parte de los anexos.

3.4. Vídeo

Se ha simulado sobre un escenario simple una transmisión de video MPEG-4 generada a partir de una fuente de video sintética. A diferencia de la transmisión CBR para una transmisión de vídeo el tráfico tiene un comportamiento a ráfagas y los paquetes enviados no tienen por qué ser del mismo tamaño.

3.4.1. Un enlace con fuente de vídeo

Fig. 3.44 Escenario de la simulación

Características de la simulación

Protocolo: UDP
Tipos de datos: Vídeo MPEG-4
Tamaño trama: Variable
Payload: 1500 bytes

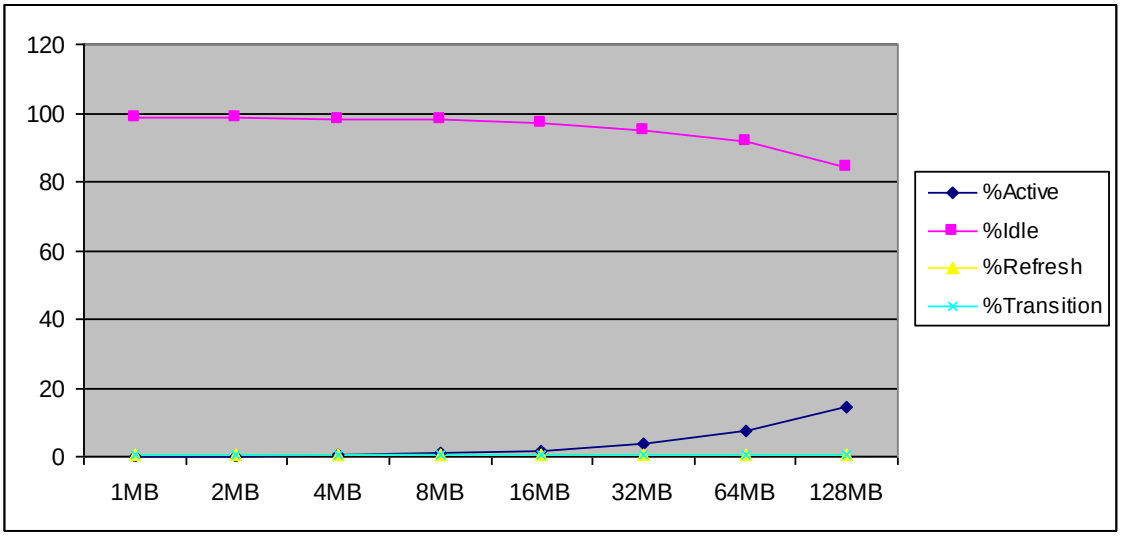

Fig. 3.45 Porcentaje de tiempo en cada estado en función de la tasa media de vídeo

De forma similar al caso de una transmisión CBR sobre UDP, para todas las tasas simuladas la utilización del enlace es inferior al 20%. Independientemente del comportamiento a ráfagas de la fuente la utilización es lo bastante baja para que el estado predominante sea el de LPI y por tanto el ahorro potencial de energía eléctrica sea elevado.

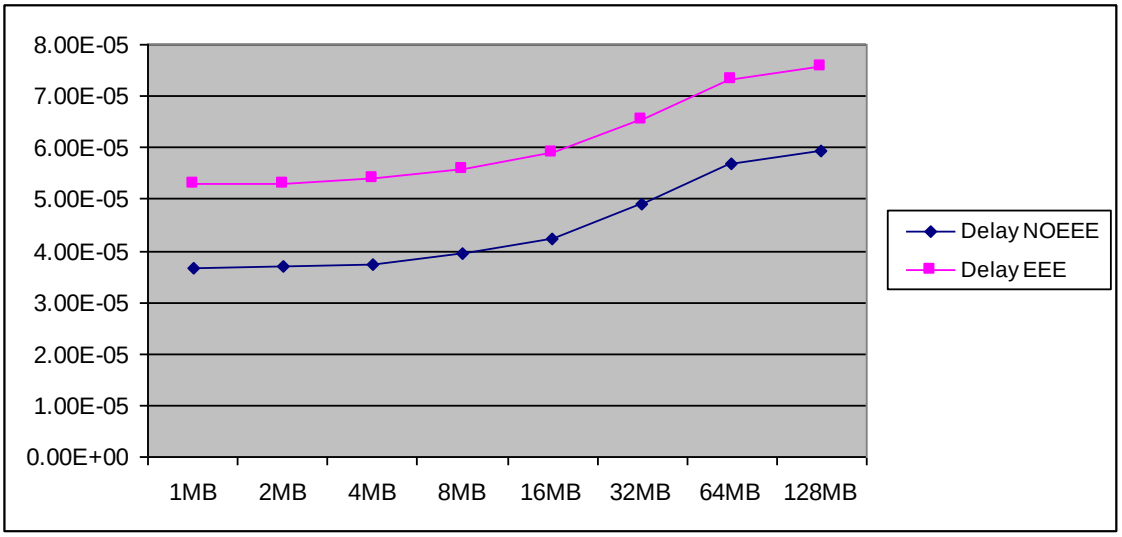

Fig. 3.46 Delay en función de la tasa media de vídeo

El delay de extremo a extremo de la transmisión se mantiene para todas las tasas un tiempo de Wake superior al una transmisión sobre Ethernet clásico. Esto se debe a que el delay extremo a extremo ha sido medido desde el momento en que el paquete se genera en el nodo de origen hasta el momento en que termina de ser recibido por el nodo de destino. La siguiente tabla ilustra un ejemplo de lo que ocurre con los paquetes de la ráfaga (eliminando en los datos de la tabla los delays adicionales mencionados en el apartado 2.7 para facilitar la comprensión y explicación de la tabla).

Tabla 3.1. Ejemplo de vida de los tres primeros paquetes de una ráfaga

ID del paquete	0	1	2
Instante de generación en el nodo 0	0	12 µs	24 µs
Instante de entrada en la cola de salida del nodo 0	0 µs	12 µs	24 µs
Instante de salida de la cola de salida del nodo 0	0 µs	17 µs	17 µs
Instante de recepción en el nodo 1	32 µs	44 µs	56 µs

1) Se observa que el primer paquete se genera en el nodo 0 en el instante 0 y pasa a la cola de salida. Al ser el primer paquete encuentra la cola vacía y tarda en ser transmitido 32 µs (un tiempo de Wake de 17 µs, un tiempo de propagación de 10 µs y un tiempo de transmisión de 12 µs).

2) El segundo paquete de la ráfaga se genera en el nodo 0 en el instante 12 µs y pasa a la cola de salida, pero a diferencia del primero se encuentra con que el medio está ocupado con la transmisión del paquete anterior y tiene que esperar en la cola durante 17 µs. Después tarda otros 22 µs (un tiempo de propagación de 10 µs y un tiempo de transmisión de 12 µs) en ser recibido por completo por el nodo 1.

3) Lo mismo ocurre con el tercer paquete que también tiene que esperar en la cola de salida durante 17 µs y con los todos los paquetes posteriores de la ráfaga.

Por tanto ya sea por esperar el tiempo de Wake o por esperar que el medio este libre tras completarse la transmisión del paquete anterior, todos los paquetes de la ráfaga sufren un delay adicional de un tiempo de Wake de 17 µs.

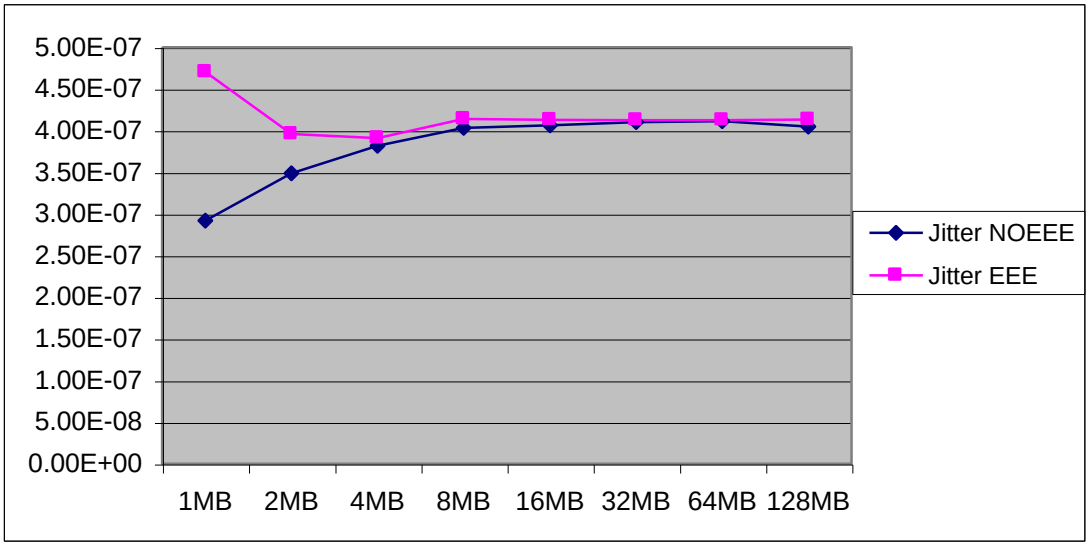

Fig. 3.47 Delay en función de la tasa media de vídeo

Como hemos visto en el análisis del delay, todos los paquetes de la ráfaga sufren el delay equivalente a una transición Wake. Por tanto el jitter introducido por EEE es mínimo. El jitter resultante de la transmisión de video sobre una LAN con EEE es prácticamente el mismo que sobre una LAN convencional, con variaciones inferiores a 1 µs en el peor de los casos.

CAPÍTULO 4. CONCLUSIONES Y LÍNEAS FUTURAS

4.1. Conclusiones

A lo largo de este trabajo se ha realizado un análisis del protocolo IEEE 802.3az (Energy Efficient Ethernet) tanto en términos de ahorro de energía como de efecto sobre el patrón de tráfico de las redes Ethernet. Como conclusión podemos afirmar que en términos generales EEE cumple con su propósito, obtiene un ahorro significativo de energía eléctrica a costa de unos efectos sobre el tráfico que se vuelven prácticamente despreciables para delays de extremo a extremo superiores a 1 ms.

No obstante estos efectos negativos sobre el tráfico si pueden llegar a ser determinantes en condiciones concretas.

Se ha identificado como uno de esos casos el de una transmisión FTP utilizando TCP-Vegas entre nodos pertenecientes a una misma red de área local, donde la variación del RTT ocasionada por EEE causa una notable reducción de la velocidad efectiva de la transferencia de archivos. También se ha comprobado que para las otras dos variantes de TCP puestas a prueba, TCP-Linux y TCP-Reno, en las cuales el tamaño de la ventana de transmisión no depende del RTT el efecto de EEE es poco significativo.

Hemos puesto a prueba la figura de mérito de Idle por velocidad efectiva e identificado las zonas óptimas de trabajo en las que el compromiso entre ahorro de energía y velocidad de la transmisión de datos adquiere valores máximos para una serie de escenarios de uso cotidiano. Hemos comprobado que en las transmisiones sobre UDP obtenemos los ahorros de energía más significativos para valores bajos de utilización del enlace y establecido una tasa para 1000Base-T a partir de la cual ya no se produce ahorro alguno de energía.

También hemos visto el efecto de EEE sobre transmisiones con requerimientos de tiempo real y comportamiento a ráfagas como la fuente de video MPEG-4 y averiguado que en contra de lo que puede parecer intuitivamente, no necesariamente se genera un jitter adicional elevado.

Adicionalmente, se han identificado y documentado algunos efectos introducidos por el simulador que son importantes en el momento de analizar los resultados. Esto ha llevado cierto tiempo, ya que supone hacer análisis tediosos a nivel de paquete, identificando la evolución de sus estados y tiempos, pero que consideramos bien empleado ya que ha servido para validar las simulaciones y estar razonablemente seguros de los resultados.

4.2. Líneas futuras

De cara a posteriores estudios en los que profundizar sobre el efecto de EEE sería aconsejable hacer simulaciones con múltiples fuentes de tráfico independientes entre sí sobre el mismo enlace. Esto sería especialmente interesante para tráfico de vídeo, dadas sus peculiares características.

También seria útil una mayor profundización del estudio del protocolo TCP sobre EEE mediante un análisis más detallado del efecto de los tramos EEE no consecutivos dentro de una red de área local y la evolución del efecto de los mecanismos de control de congestión de TCP-Vegas sobre más de un tramo EEE.

En cuanto a posibles vías de estudio posteriores también quedaría contrastar los resultados simulados con medidas reales sobre tarjetas de red físicas que implementen EEE, algunas de las cuales ya se encuentran actualmente en el mercado. En la EETAC disponemos de switches EEE que implementan el protocolo IEEE 802.3az, concretamente dos modelos de Zyxel de la serie GS1100 [24] y aparatos de medida del consumo de energía como el Agilent N6705A [35].

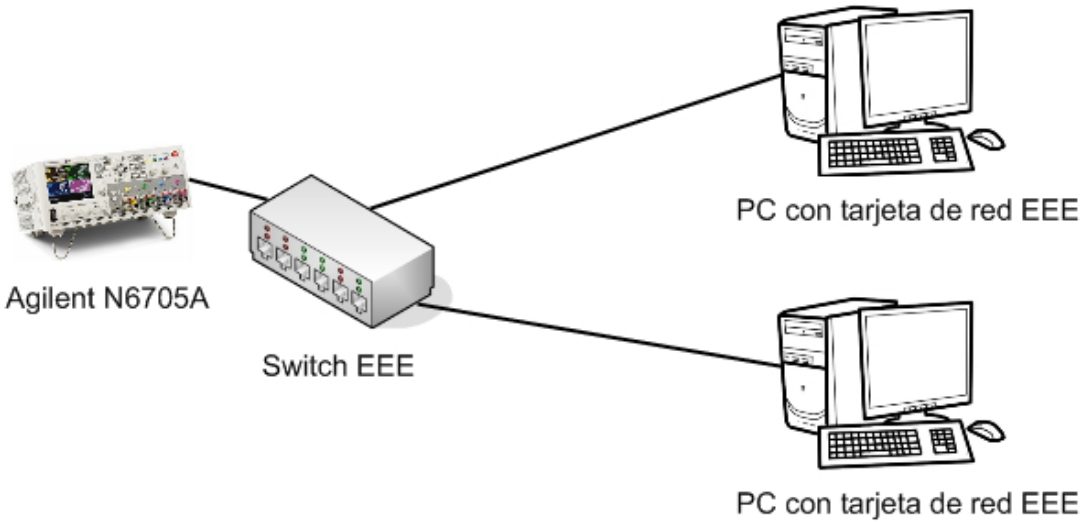

Fig. 4.1 Posible escenario de medidas reales

El campo de estudio de ahorro de energía va cobrando importancia con el tiempo. Simplemente por las mejoras en los componentes fabricados e independientemente de la implementación de protocolos para reducir el gasto energético, el consumo de las tarjetas Ethernet es cada vez menor. Por supuesto las tarjetas de red que implementen EEE con los años lograrán porcentajes de consumo en el estado de LPI cada vez más bajos.

El propio estándar es susceptible de ser mejorado a medida que la tecnología lo permita estableciendo tiempos de Wake, Sleep y Refresh más cortos. Cabe

destacar que convendría que el tiempo de Sleep para 1000Base-T fuese el primero en ser reducido por ser el más elevado con respecto a los demás en proporción a la velocidad de transmisión.

4.3. Aspectos ambientales

Aunque el desarrollo del TFC y las actividades realizadas (estudio teórico, simulaciones) no han tenido implicaciones medioambientales, consideramos que el TFC está ambientalizado, en el sentido de que aporta datos y conclusiones relacionados con el consumo de energía en redes de ordenadores.

Las infraestructuras de redes de telecomunicaciones van cobrando lenta pero inexorablemente importancia en lo referente al porcentaje de consumo de energía con respecto al consumo global. Las redes de área local no son una excepción, el número de usuarios domésticos que utilizan redes de área local ha aumentado enormemente en los últimos años y la huella dejada por el consumo de estas también ha aumentado en la misma medida.

Del mismo modo que el ahorro de energía comporta un beneficio económico también implica una reducción en las emisiones de CO_2. Es por ello que el uso de protocolos como EEE tiene un efecto evidente en la reducción del impacto medioambiental.

GLOSARIO

CBR	**Constant Bit Rate**
CRC	**Comprobación de Redundancia Cíclica**
EEE	**Energy Efficient Ethernet**
FTP	**File Transfer Protocol**
ID	**Identificador**
IEEE	**Institute of Electrical and Electronic Engineers**
LLDP	**Link Layer Discovery Protocol**
LPI	**Low Power Idle**
MPEG	**Moving Picture Experts Group**
MTU	**Maximum Transfer Unit**
Nam	**Network Animator**
NS-2	**Network Simulator 2**
OTcl	**Object-oriented Tool Command Language**
RTT	**Round Trip Time**
SFD	**Start Frame Delimiter**
Tq	**Quiet time**
Tr	**Refresh time**
Ts	**Sleep time**
Tw	**Wake time**
TCP	**Transmission Control Protocol**
UDP	**User Datagram Protocol**

CAPÍTULO 5. BIBLIOGRAFÍA

[1]. Ezponda Aycart, J., “Estudi i simulacions de l’estàndart Energy-Efficient-Ethernet”, Treball Fi de Carrera, Escola d’Enginyeria de Telecomunicacions i Aeroespacial de Castellefels 12 de Septiembre de 2011. https://upcommons.upc.edu/pfc/handle/2099.1/12889

[2]. Lightwave, “IEEE ratifies new 802.3az standart to reduce network energy footprint”, http://www.lightwaveonline.com/education/news/IEEE-ratifies-new-8023az-standard-to-reduce-network-energy-footprint--104392129.html

[3]. “IEEE 802.3az Energy Efficient Ethernet Task Force”; http://grouper.ieee.org/groups/802/3/az

[4]. Chistensen, K - Reviriego, P - Nordman, B - Bennet, M - Mostowfi, M - Maestro, J.A, "IEEE 802.3az: "The Road to Energy Efficient Ethernet", IEEE Communications Magazine, Nov. 2010

[5]. “802.3az Draft 2.3 Part 3: Carrier Sense Multiple Acces with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications”, 27 October 2010

[6]. Reviriego, P., Hernández J. A., Larrabeiti, D., and Maestro, J.A. "Performance Evaluation of Efficient Ethernet", IEEE Communications Letters, Vol.13, Num. 9, September 2009

[7]. Reviriego, P., Hernández J. A., Larrabeiti, D., and Maestro, J.A. " Increasing the MTU size for Energy Efficiency in Ethernet", IEEE INTERNET COMPUTING, July/August 2010

[8]. Reviriego P. et al., “Burst Transmission in Energy Efficient Ethernet,” IEEE Internet Comp., vol. 14, no. 4, July/Aug. 2010, pp. 50–57.

[9]. Reviriego P - Maestro, J.A, " An Initial Evaluation of Energy Efficient Ethernet ", IEEE Communications Letters, Vol.15, Num. 5, May 2011

[10]. “Nsnam”, http://nsnam.isi.edu/nsnam/index.php/Main_Page

[11]. Greis M, “NS-2 tutorial”. http://www.isi.edu/nsnam/ns/tutorial/

[12]. Issariyakul, T. - Hossain, E., "*Introduction to Network Simulator NS2*", Springer, 2009.

[13]. Steffen Reidt, "What to do with NS-2", September 2007, http://www.isg.rhul.ac.uk/~prai175/ISGStudentSem07/NS2.pdf

[14]. Chih-Heng, Ke, “How to measure packet loss rate, jitter, and end-to-end delay for UDP-based applications?”, http://hpds.ee.ncku.edu.tw/~smallko/ns2/tool_en.htm

[15]. Matrawy A, “MPEG4”, http://www.sce.carleton.ca/~amatrawy/mpeg4/

[16]. Bharathi Subramanian, "NS2 Architecture" http://bharathi.posterous.com/ns2-architecture

[17]. OTcl, http://otcl-tclcl.sourceforge.net/otcl/

[18]. Macian Perez A S, Reviriero Vasolla P, “Módulo EEE para NS-2”

[19]. “Delay Tolerant Networking Research Group”, http://www.dtnrg.org/wiki

[20]. Kinder L, “Simulation Tools”, http://code.n4c.eu/code/__simulation-tools/archive/tip.__zip

[21]. “The One Simulator”, http://code.n4c.eu/code/__simulation-tools/archive/tip.__zip

[22]. “DTNSim2”, http://watwire.uwaterloo.ca/DTN/sim/

[23]. Eriksson H, Jönsson P, “Nsdtn”, http://www.illuvatar.nu/ns-dtn/code/

[24]. “Switch Zyxel GS1100-16”, http://www.zyxel.com/products_services/es1100_series.shtml?t=p

[25]. Chriestensen k, “Green Networks: Reducing the energy consumption of networks”.

[26]. Wikipedia, “Link Layer Discovery Protocol”, http://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol

[27]. Wikipedia, “Energy Efficient Ethernet”, http://en.wikipedia.org/wiki/802.3az

[28]. Wikipedia, “Ethernet”, http://es.wikipedia.org/wiki/Ethernet

[29]. Wikipedia, “IEEE 802.3”, http://es.wikipedia.org/wiki/IEEE_802.3

[30]. Davis L, “Ethernet Bus”, http://www.interfacebus.com/Design_Connector_Ethernet.html

[31]. Tolosa Rodríguez L, “Gigabit Ethernet”, http://www.monografias.com/trabajos12/giga/giga.shtml

[32]. Wikipedia, “TCP Vegas”, http://es.wikipedia.org/wiki/TCP_Vegas

[33]. Wikipedia, “TCP Reno”, http://es.wikipedia.org/wiki/Reno_TCP

[34]. Wikipedia, “Implementaciones de TCP”, http://es.wikipedia.org/wiki/Implementaciones_de_TCP#Linux

[35]. Rodríguez Gillen L M, “Mesura de consum de dispositius de xarxa”, Treball Fi de Carrera, Escola d’Enginyeria de Telecomunicacions i Aeroespacial de Castellefels, 19 de Marzo de 2012, http://upcommons.upc.edu/pfc/bitstream/2099.1/14974/1/memoria.pdf

ANEXOS

5.1. Anexo 1. Modificaciones en el código del módulo EEE

```
void WiredEEEPhy::printTimes(){

	double totalTime = time_sleeping_ + time_waking_ + time_refreshing_ +
time_idle_ + time_active_;
	double transitionTime = time_sleeping_ + time_waking_ ;
	if(mac_addr()==0)
	{
	FILE *ft;
	FILE *fa;
	FILE *fi;
	FILE *fr;
	FILE *ftt;
	FILE *fat;
	FILE *fit;
	FILE *frt;
	FILE *fTTt;
	FILE *fTr;

	ft = fopen ( "ftransition.txt", "a+" );
	fa = fopen ( "factive.txt", "a+" );
	fi = fopen ( "fidle.txt", "a+" );
	fr = fopen ( "frefresh.txt", "a+" );

	ftt = fopen ( "ftransitionT.txt", "a+" );
	fat = fopen ( "factiveT.txt", "a+" );
	fit = fopen ( "fidleT.txt", "a+" );
	frt = fopen ( "frefreshT.txt", "a+" );
	fTTt = fopen ( "TT.txt", "a+" );
	fTr = fopen ( "TransT.txt", "a+" );

	fprintf(ft,"%.4lf\n",( transitionTime*100/totalTime));
	fprintf(fa,"%.4lf\n",( time_active_*100/totalTime));
	fprintf(fi,"%.4lf\n",( time_idle_*100/totalTime));
	fprintf(fr,"%.4lf\n",( time_refreshing_*100/totalTime));
	fprintf(ftt,"%.8lf\n",( transitionTime));
	fprintf(fat,"%.8lf\n",( time_active_));
	fprintf(fit,"%.8lf\n",( time_idle_));
	fprintf(frt,"%.8lf\n",( time_refreshing_));
	fprintf(fTTt,"%.8lf\n",( totalTime));
	fprintf(fTr,"%.8lf\n",( transitionTime));

	fclose ( ft );
	fclose ( fa );
	fclose ( fi );
	fclose ( fr );

	fclose ( ftt );
	fclose ( fat );
	fclose ( fit );
	fclose ( frt );
	fclose ( fTTt );
	fclose ( fTr );
	}

}
```

5.2. Anexo 2. Gráficos adicionales

5.2.1. Un enlace TCP

5.2.1.1. TCP-Linux

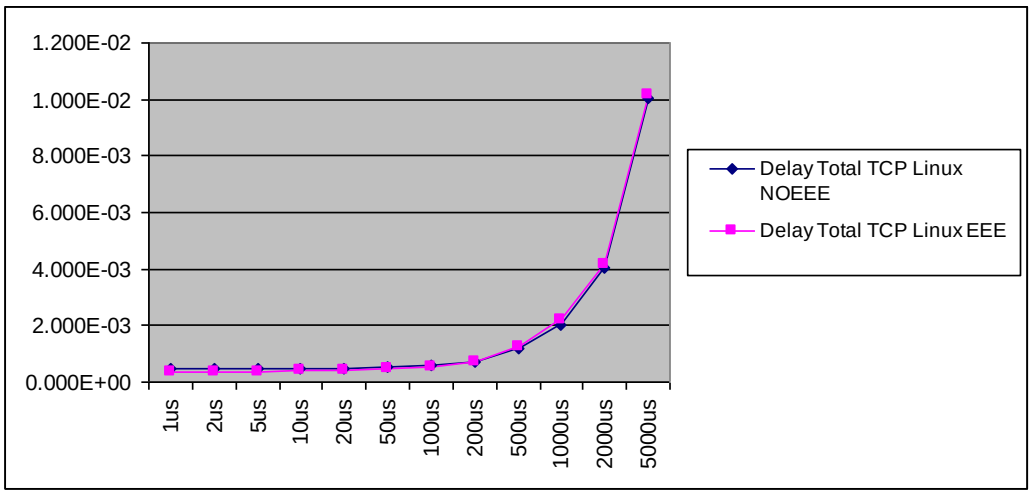

Fig. A2.1 Delay extremo a extremo en función del delay de enlace

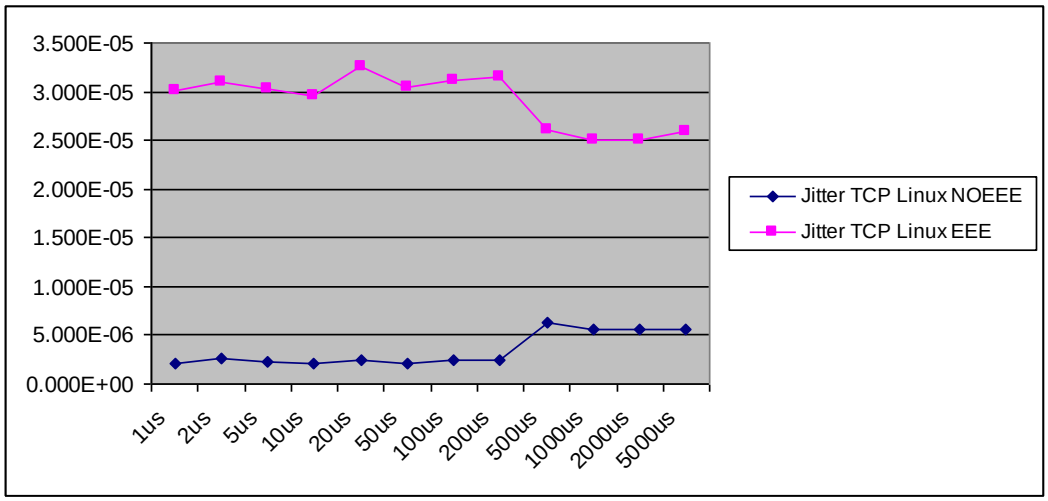

Fig. A2.2 Jitter en función del delay de enlace

5.2.1.2. TCP-Reno

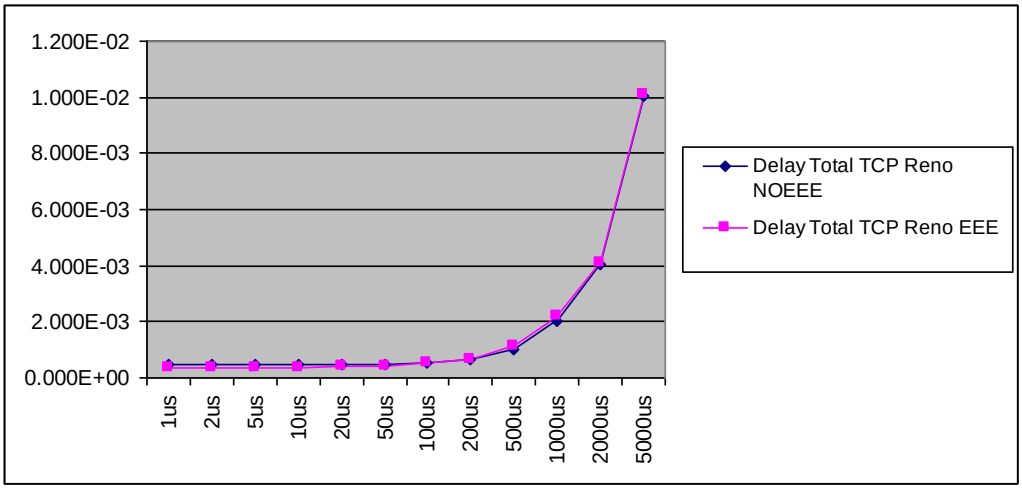

Fig. A2.3 Delay extremo a extremo en función del delay de enlace

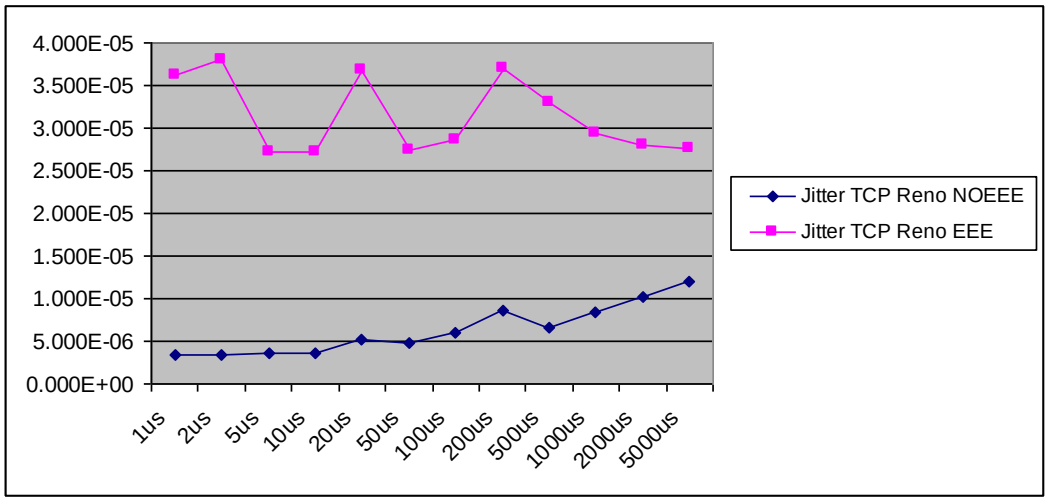

Fig. A2.4 Jitter en función del delay de enlace

5.2.1.3. TCP-Vegas

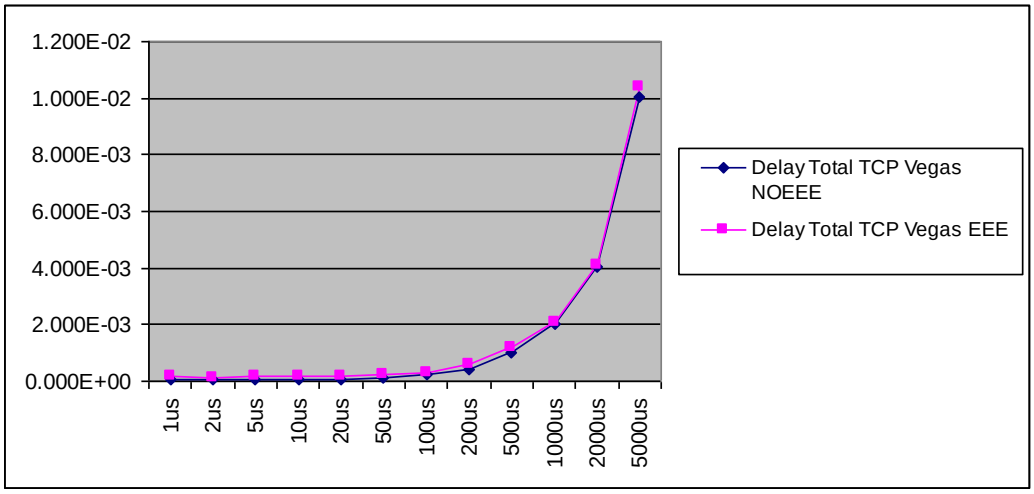

Fig. A2.5 Delay extremo a extremo en función del delay de enlace

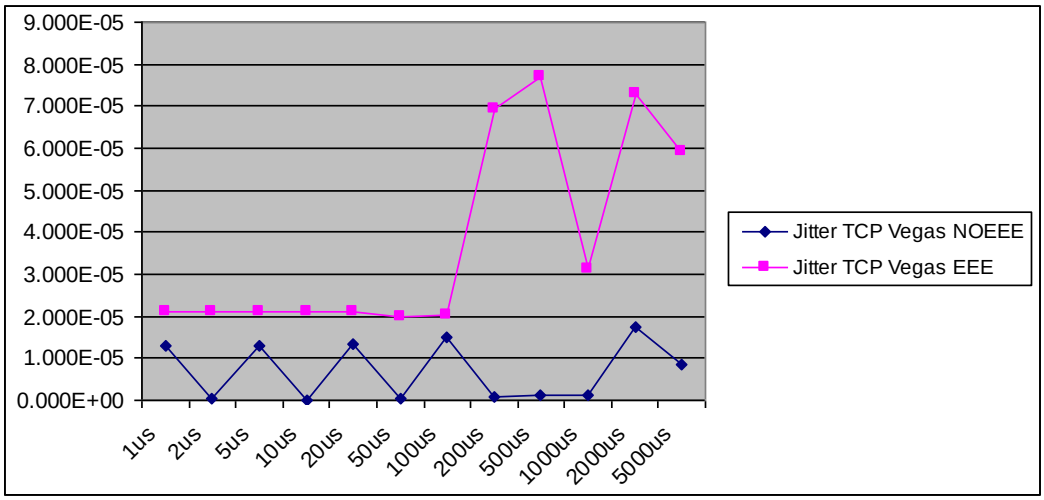

Fig. A2.6 Jitter en función del delay de enlace

5.2.1.4. TCP-Linux Ventana limitada

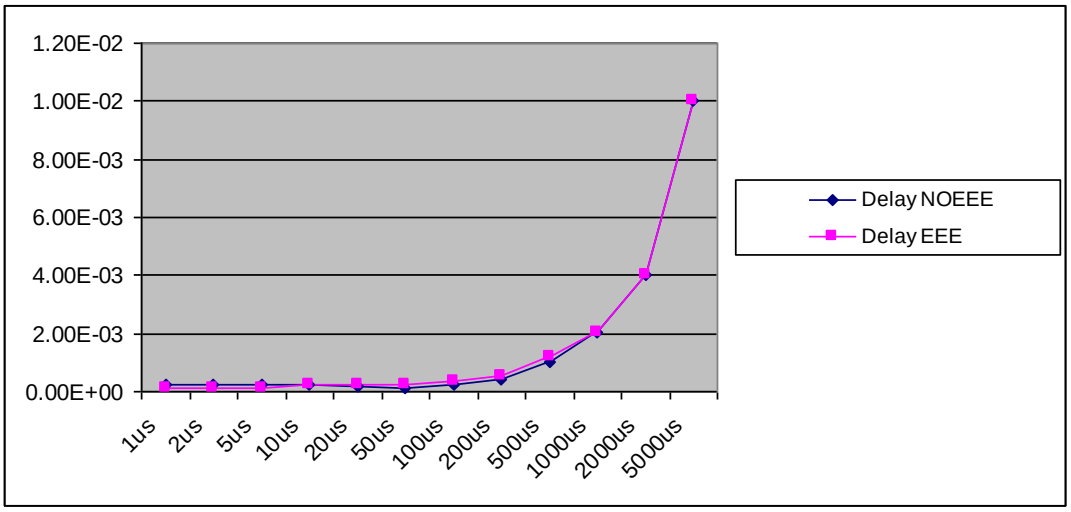

Fig. A2.7 Delay extremo a extremo en función del delay de enlace

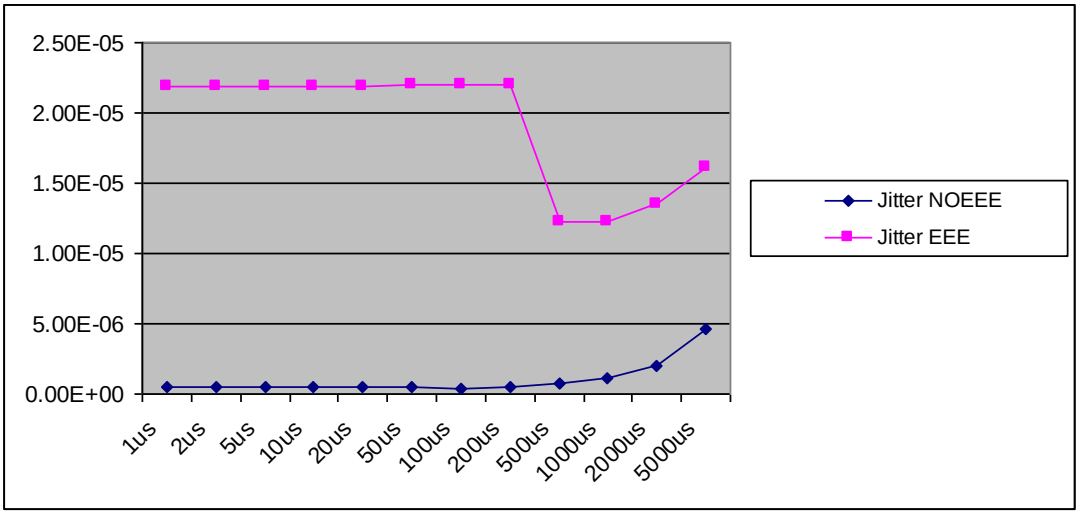

Fig. A2.8 Jitter en función del delay de enlace

5.2.2. Dos enlaces TCP

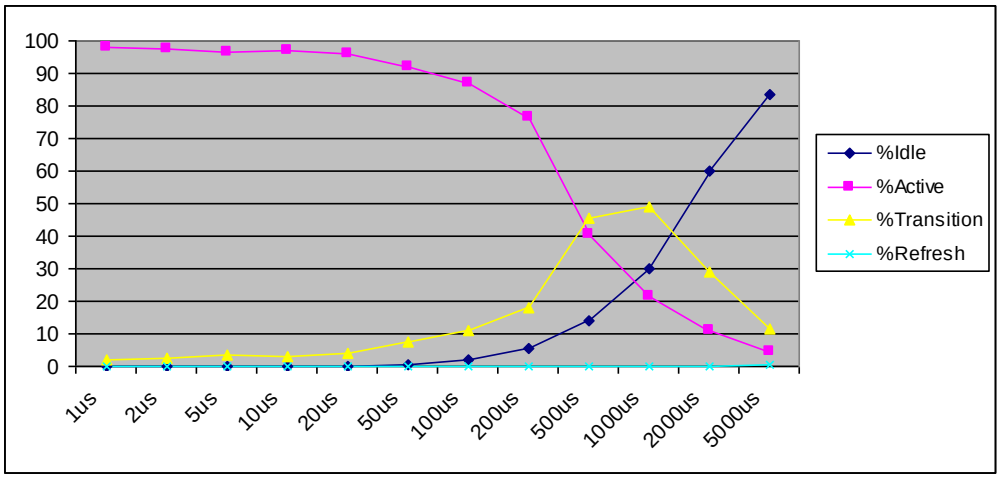

Fig. A2.9 % de tiempo en cada estado en el tramo 1

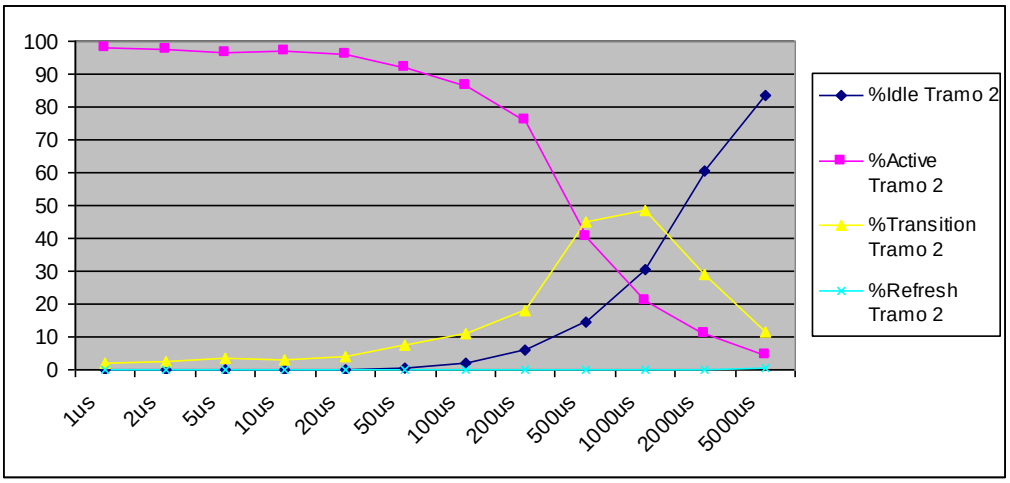

Fig. A2.10 % de tiempo en cada estado en el tramo 2

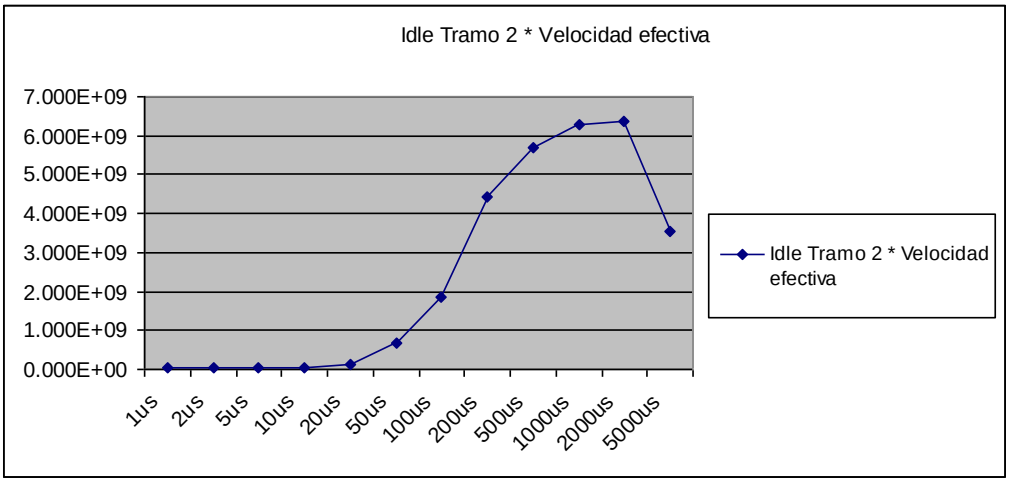

Fig. A2.11 Producto Idle tramo 2 por velocidad efectiva

5.2.2.1. TCP-Linux Ventana limitada

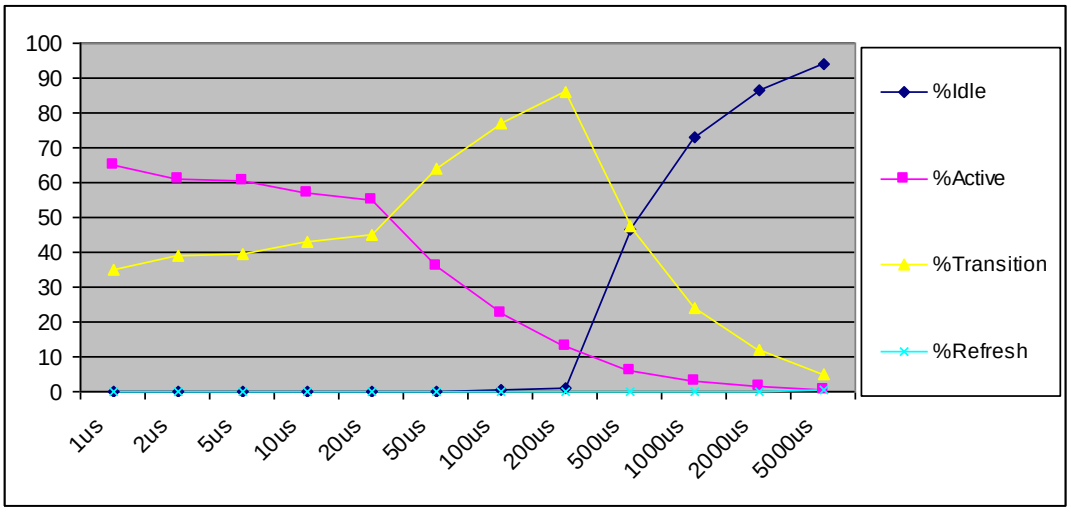

Fig. A2.12 % de tiempo en cada estado en el tramo 1

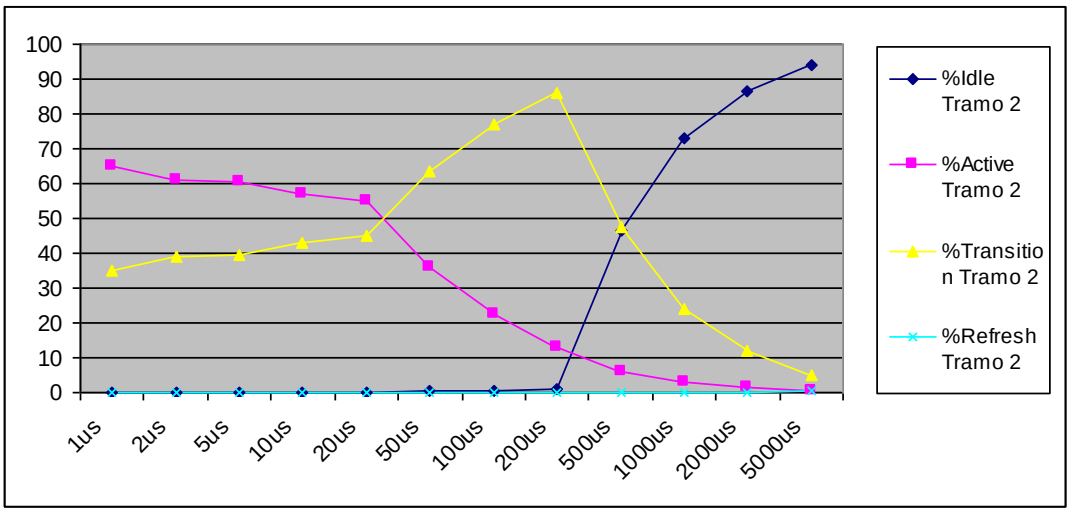

Fig. A2.13 % de tiempo en cada estado en el tramo 2

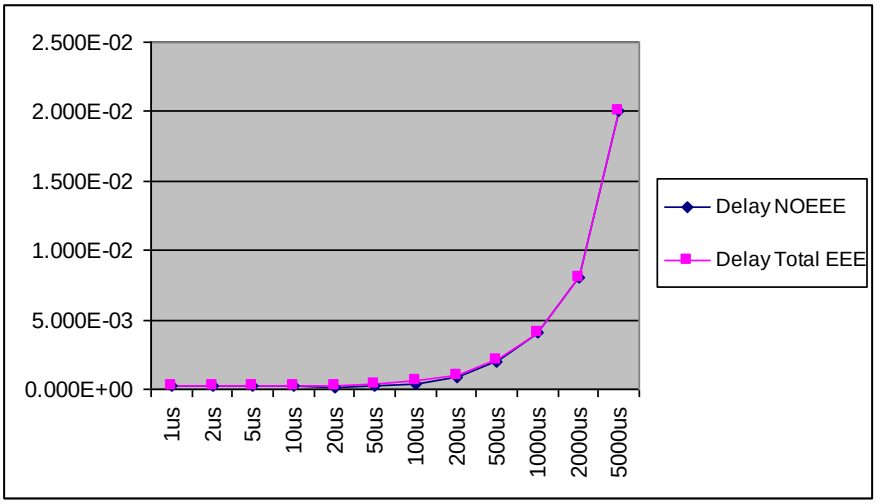

Fig. A2.14 Delay extremo a extremo en función del delay de enlace

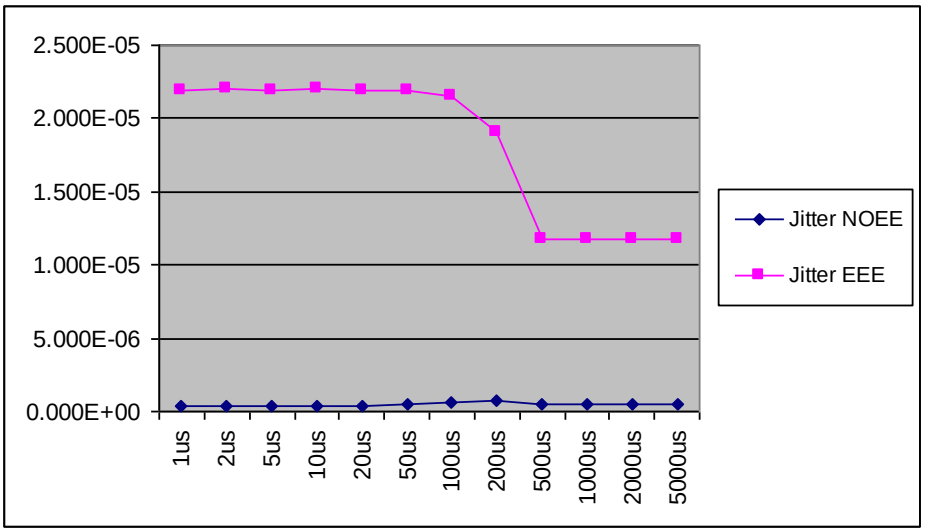

Fig. A2.15 Jitter en función del delay de enlace

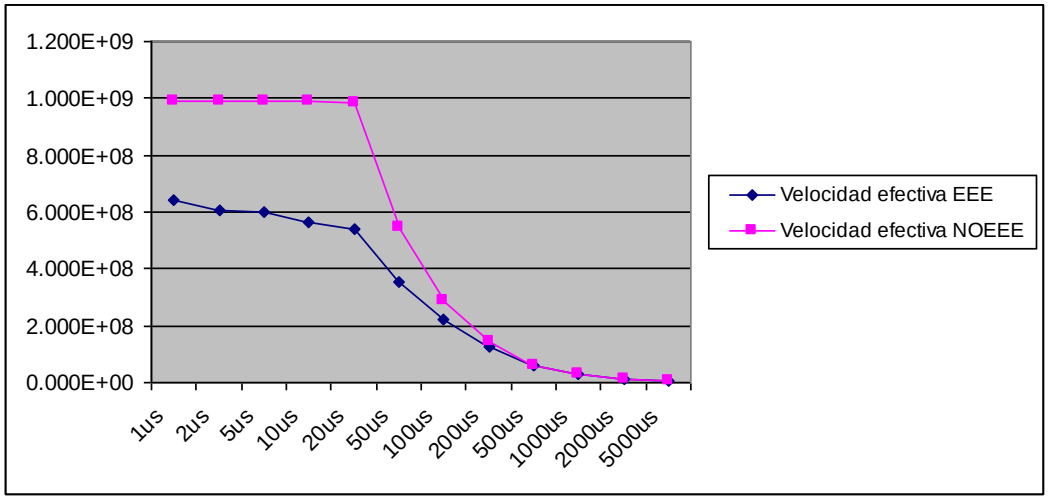

Fig. A2.16 Velocidad efectiva en función del delay de enlace

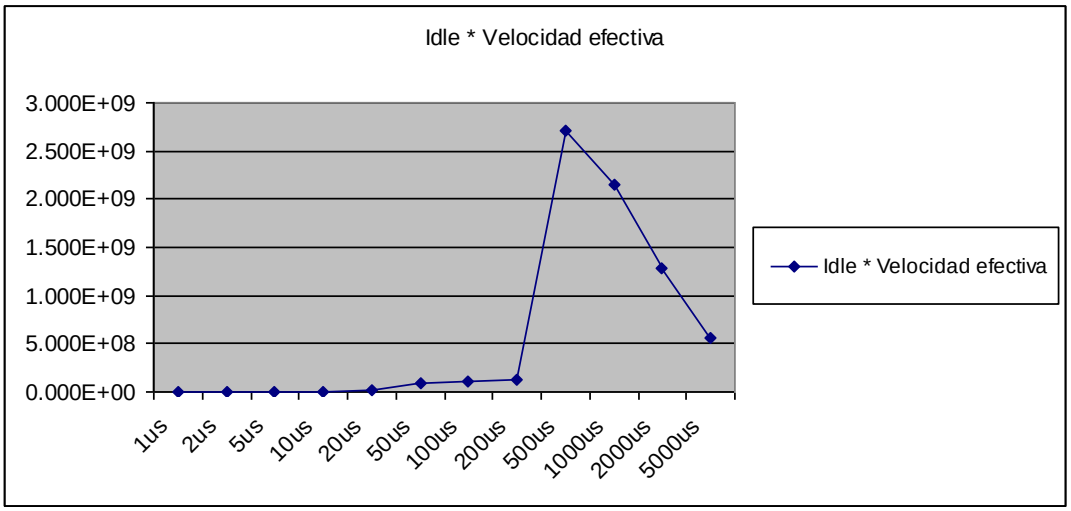

Fig. A2.17 Producto Idle tramo 1 por velocidad efectiva

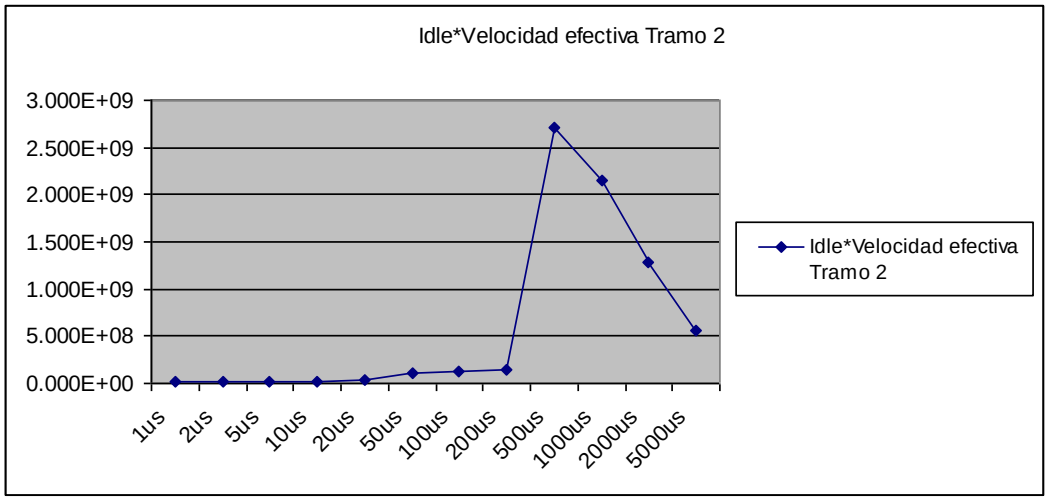

Fig. A2.18 Producto Idle tramo 2 por velocidad efectiva

5.2.3. Tres enlaces TCP

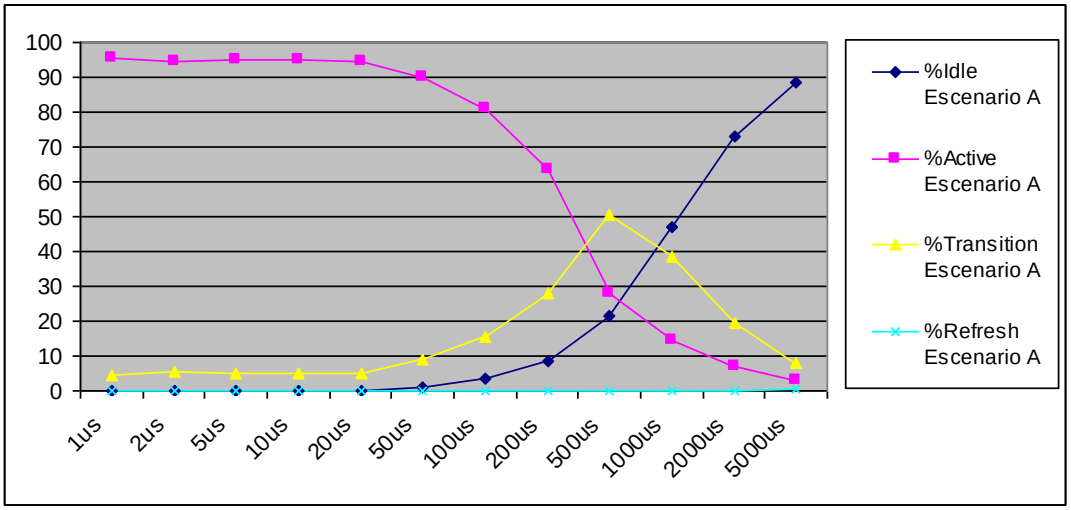

Fig. A2.19 % de tiempo en cada estado en el tramo 1

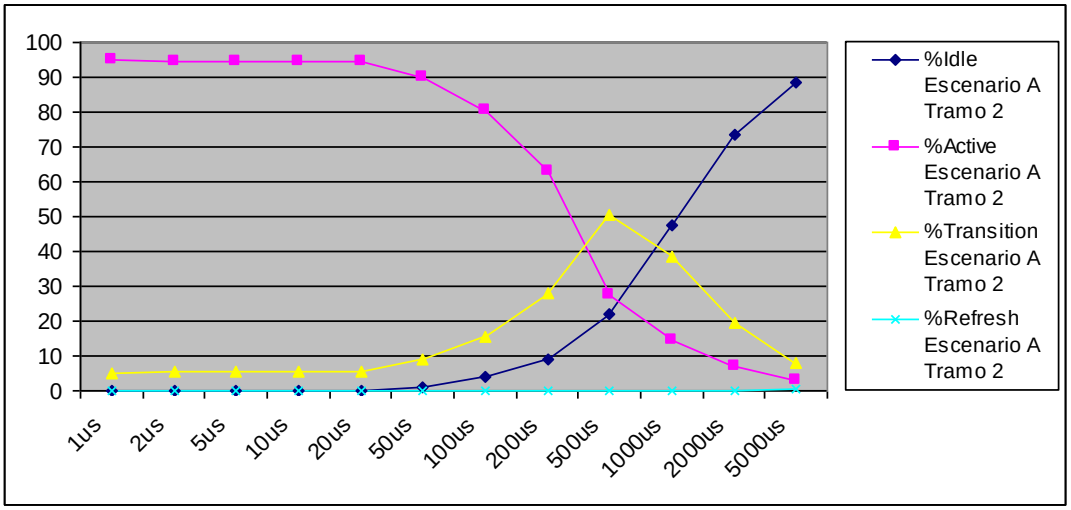

Fig. A2.20 % de tiempo en cada estado en el tramo 2

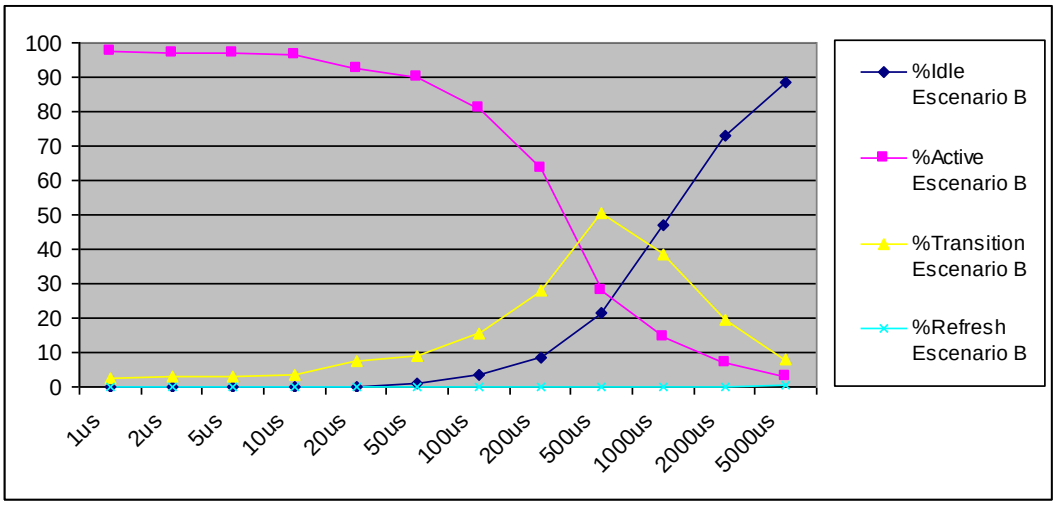

Fig. A2.21 % de tiempo en cada estado en el tramo 1

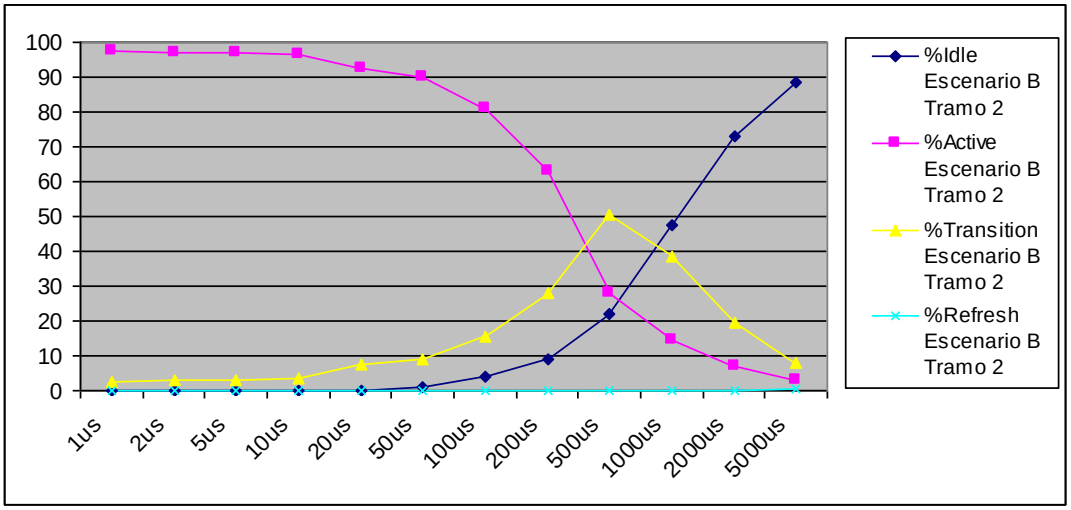

Fig. A2.22 % de tiempo en cada estado en el tramo 2

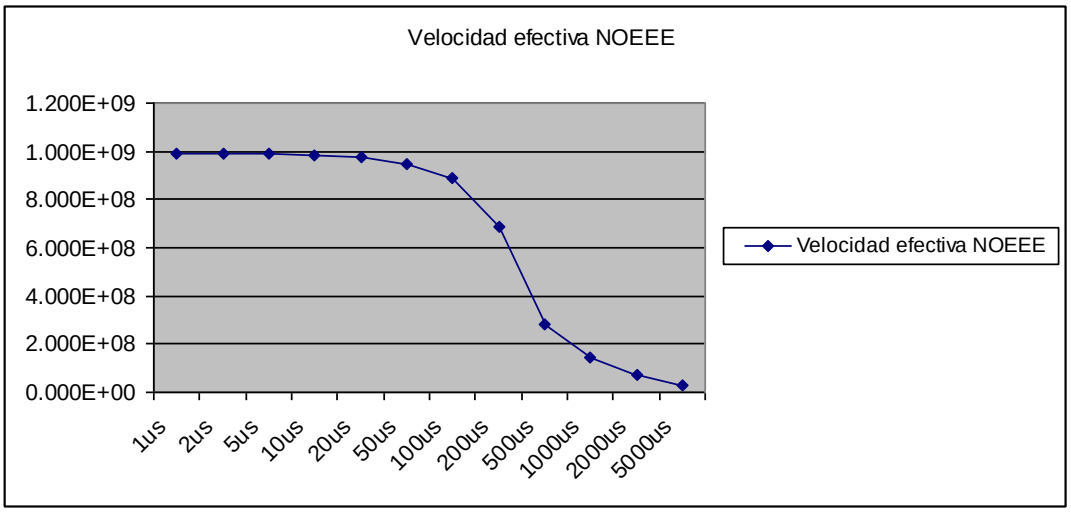

Fig. A2.23 Velocidad efectiva en función del delay de enlace

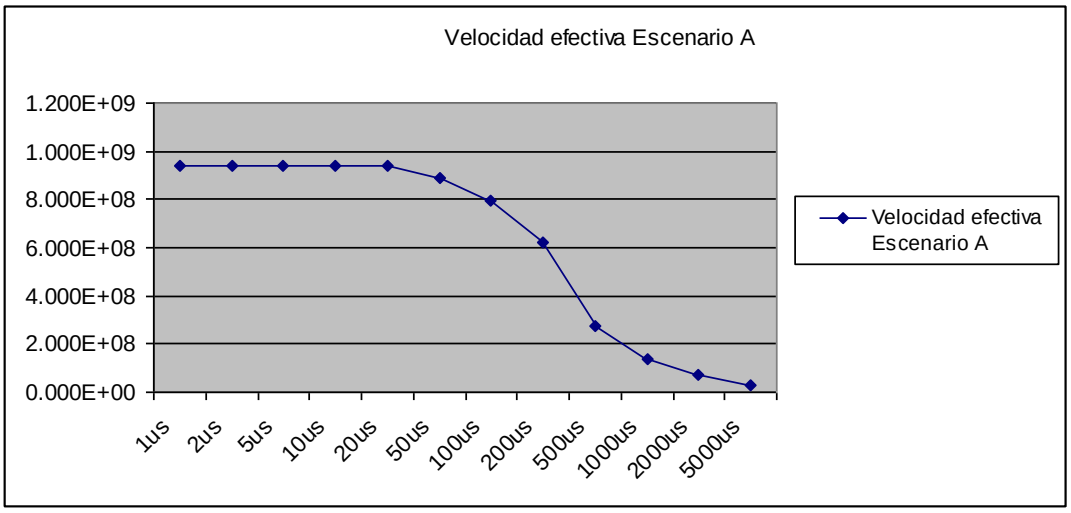

Fig. A2.24 Velocidad efectiva en función del delay de enlace

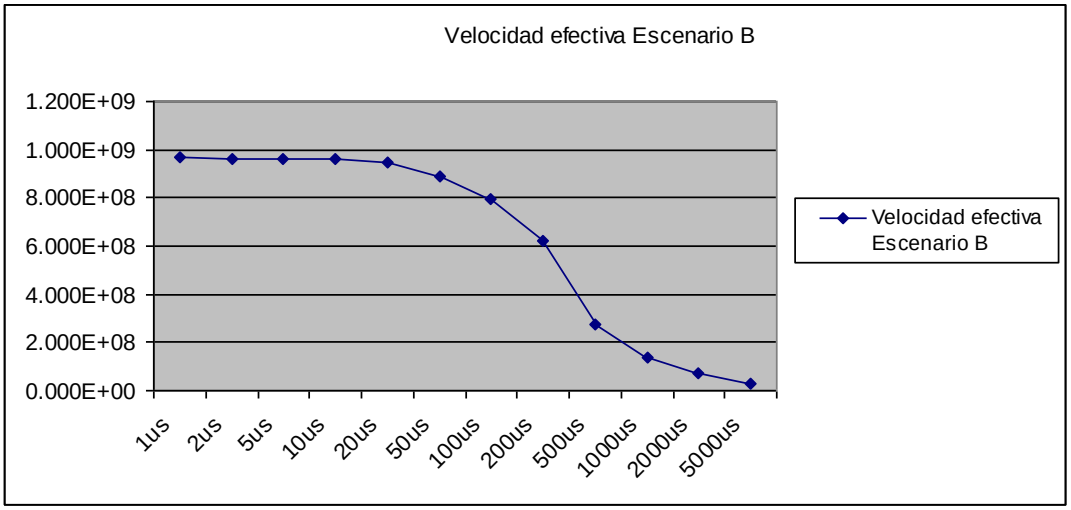

Fig. A2.25 Velocidad efectiva en función del delay de enlace

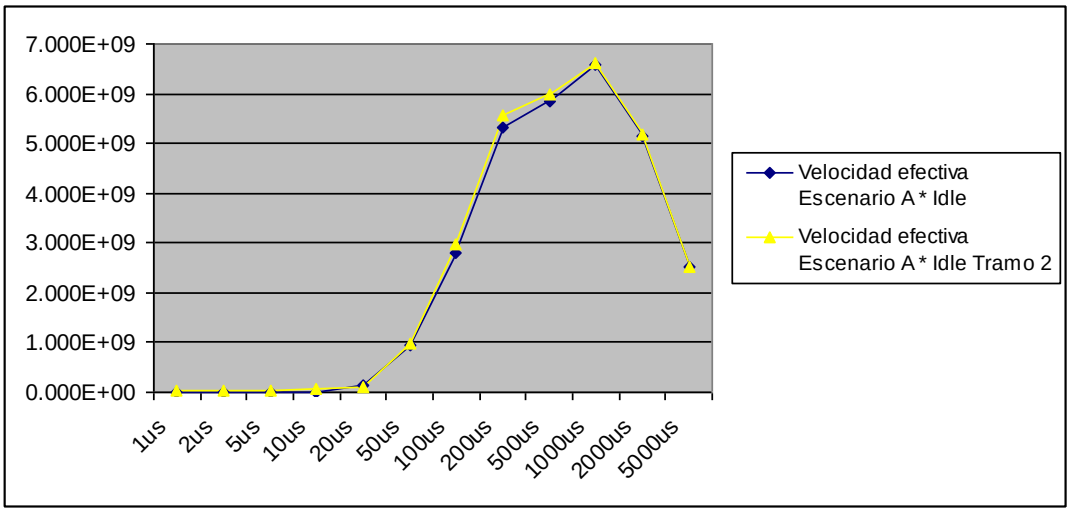

Fig. A2.26 Producto Idle por velocidad efectiva

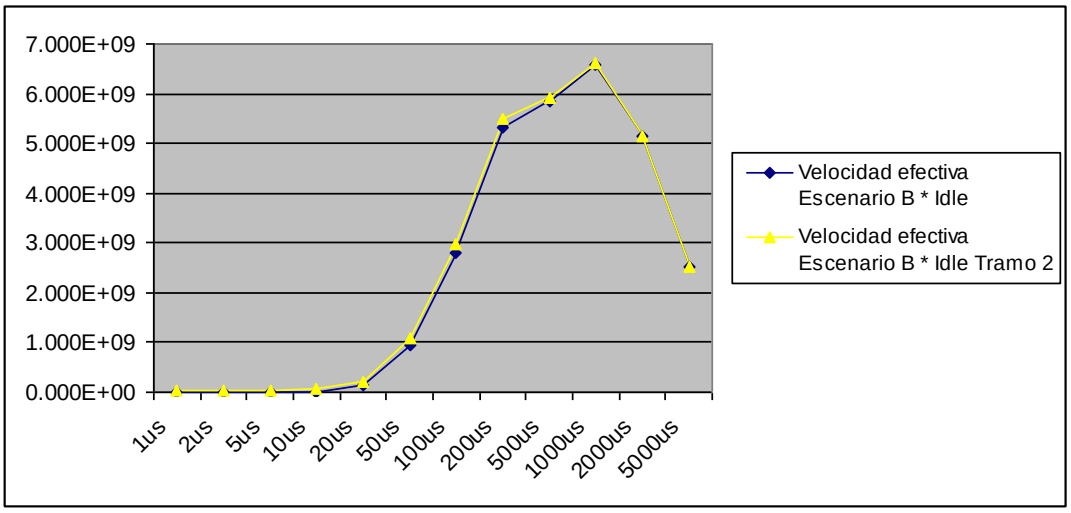

Fig. A2.27 Producto Idle por velocidad efectiva

5.2.3.1. TCP-Linux Ventana limitada

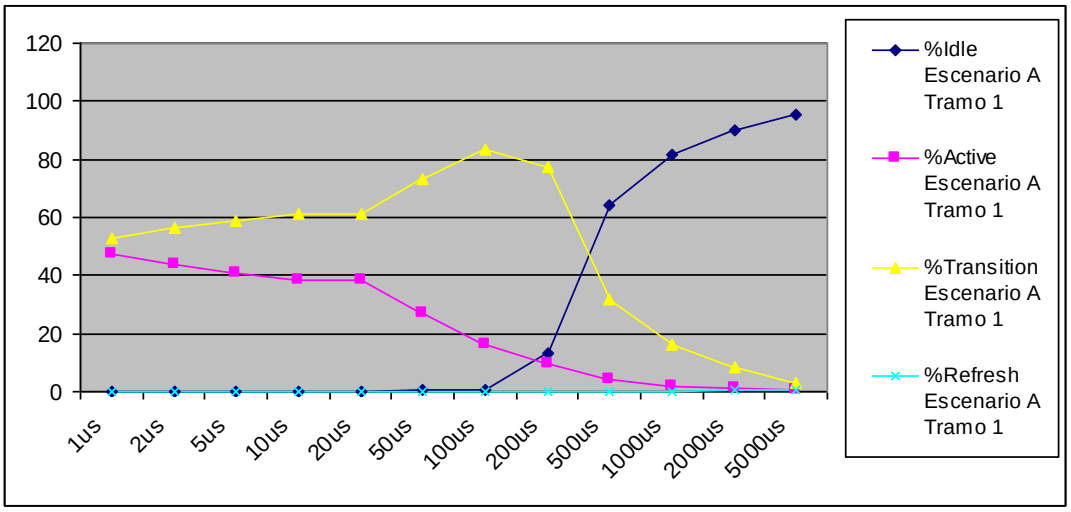

Fig. A2.28 % de tiempo en cada estado en el tramo 1

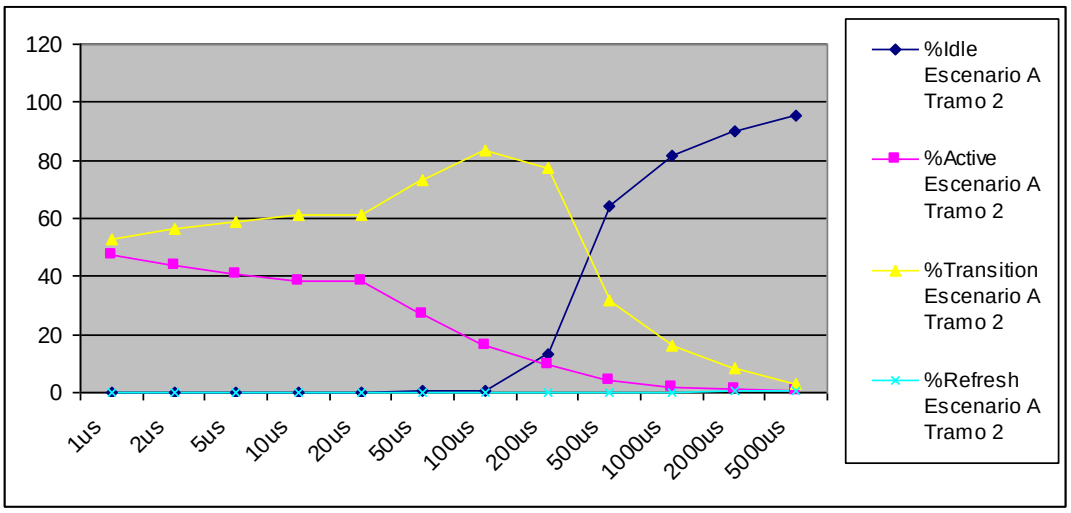

Fig. A2.29 % de tiempo en cada estado en el tramo 2

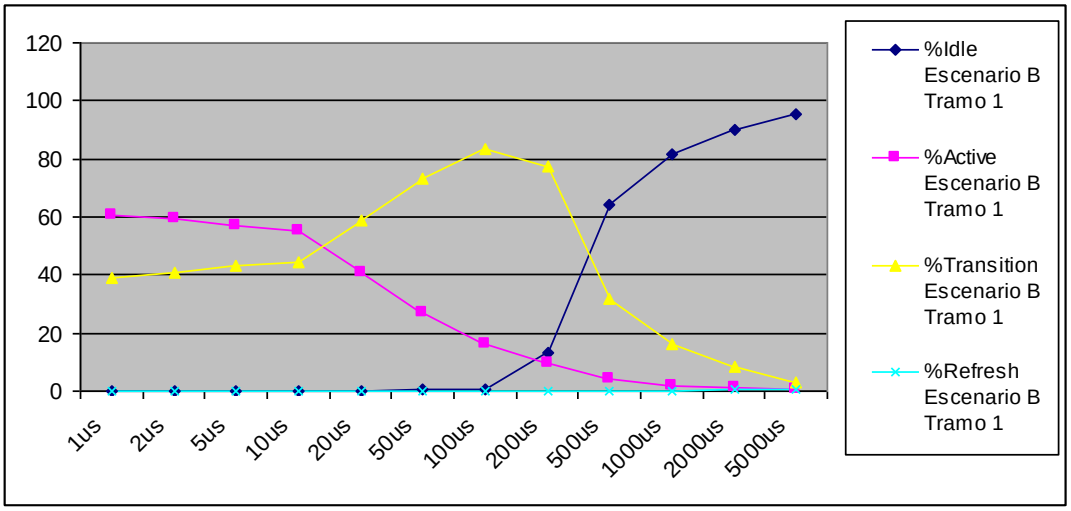

Fig. A2.30 % de tiempo en cada estado en el tramo 1

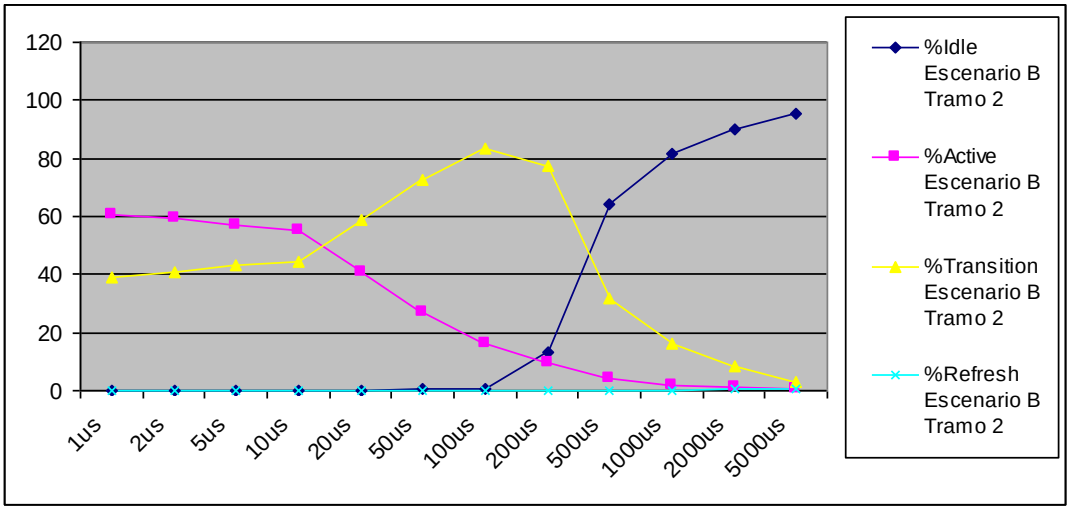

Fig. A2.31 % de tiempo en cada estado en el tramo 2

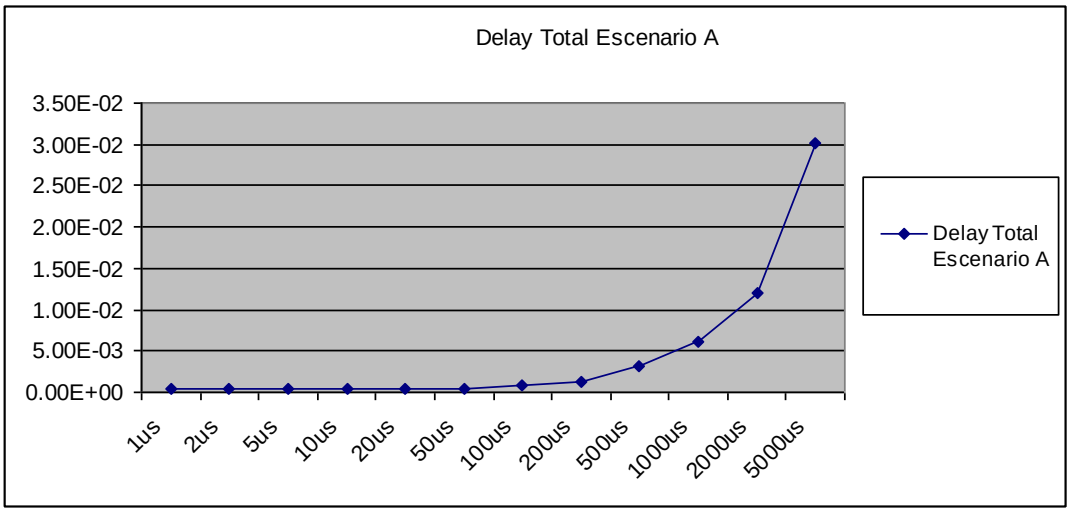

Fig. A2.32 Delay extremo a extremo en función del delay de enlace

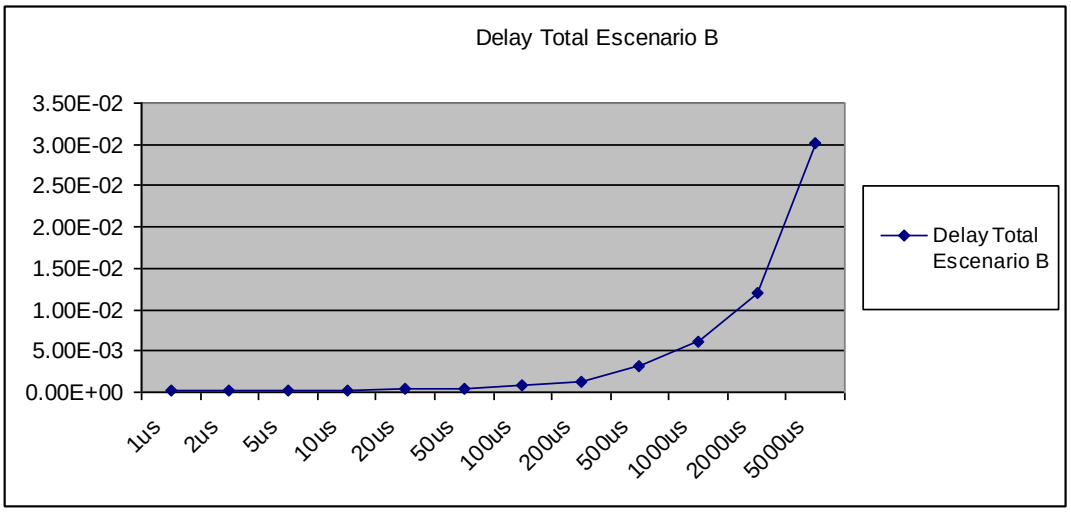

Fig. A2.33 Delay extremo a extremo en función del delay de enlace

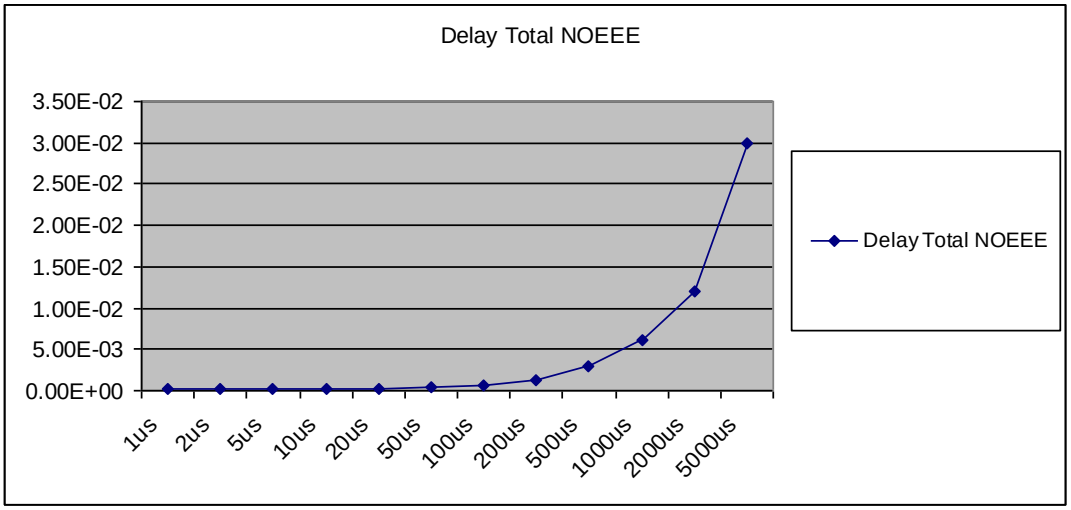

Fig. A2.34 Delay extremo a extremo en función del delay de enlace

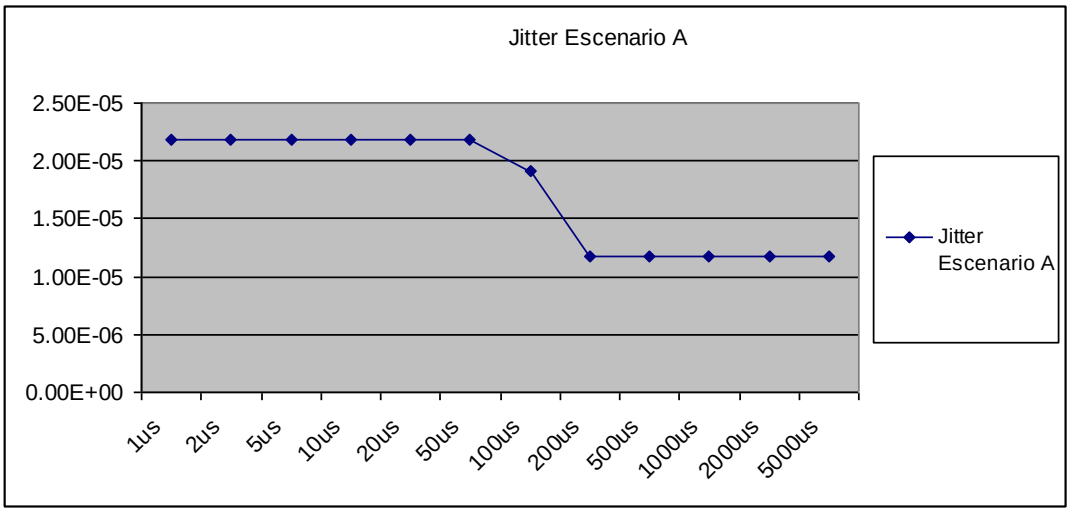

Fig. A2.35 Jitter en función del delay de enlace

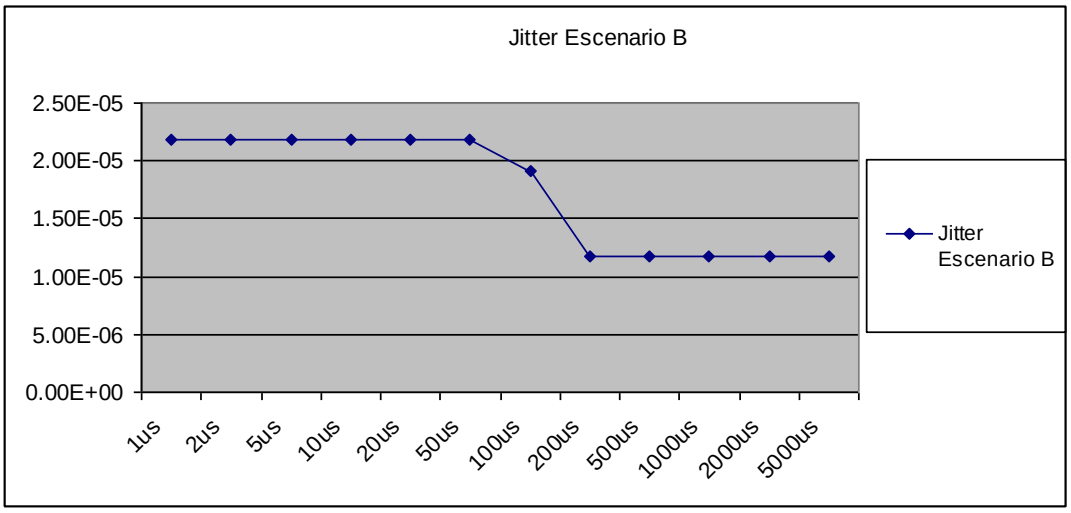

Fig. A2.36 Jitter en función del delay de enlace

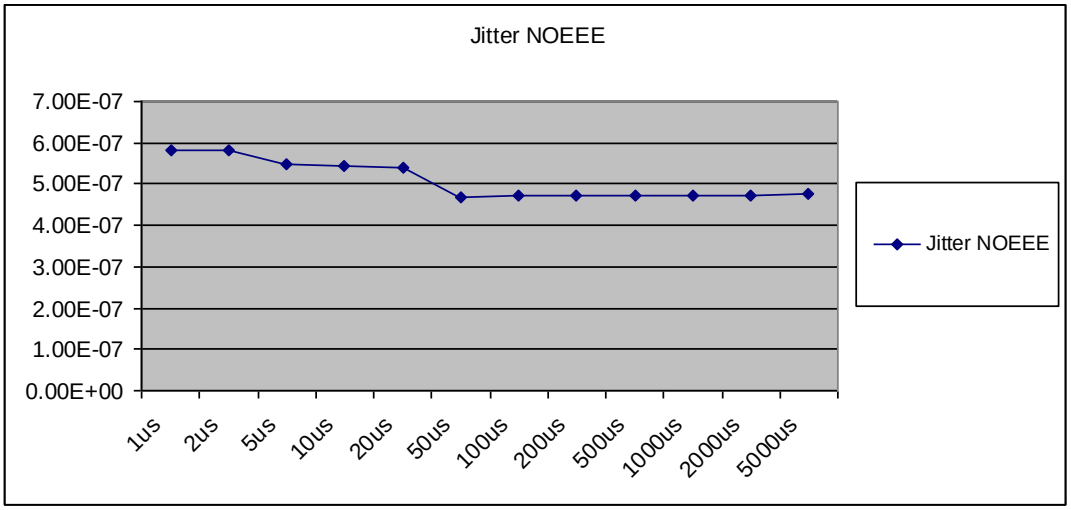

Fig. A2.37 Jitter en función del delay de enlace

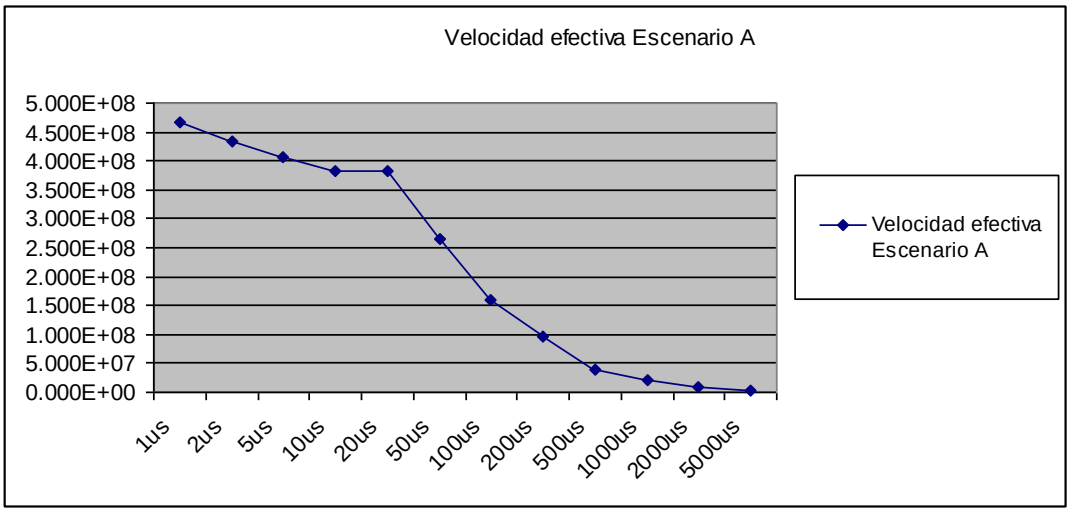

Fig. A2.38 Velocidad efectiva en función del delay de enlace

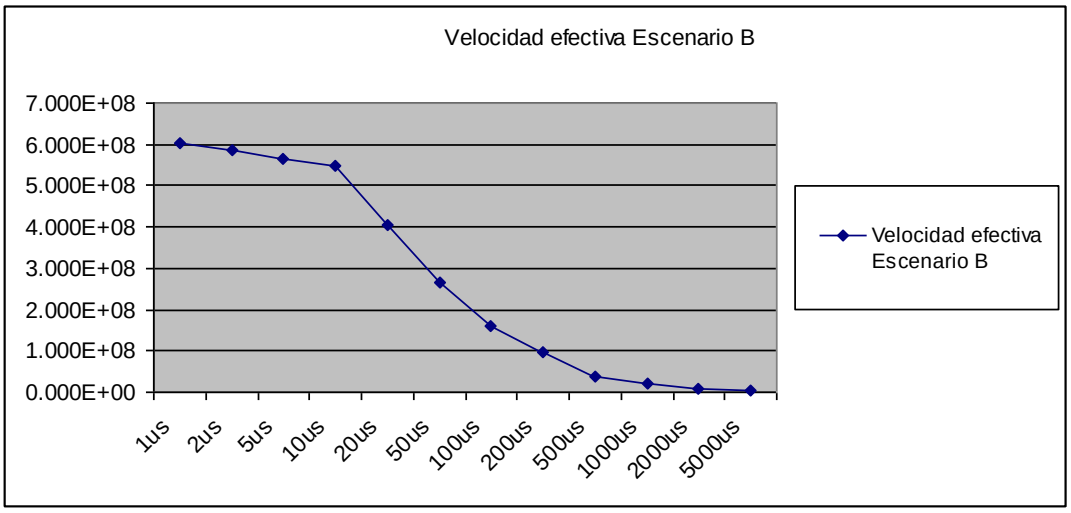

Fig. A2.39 Velocidad efectiva en función del delay de enlace

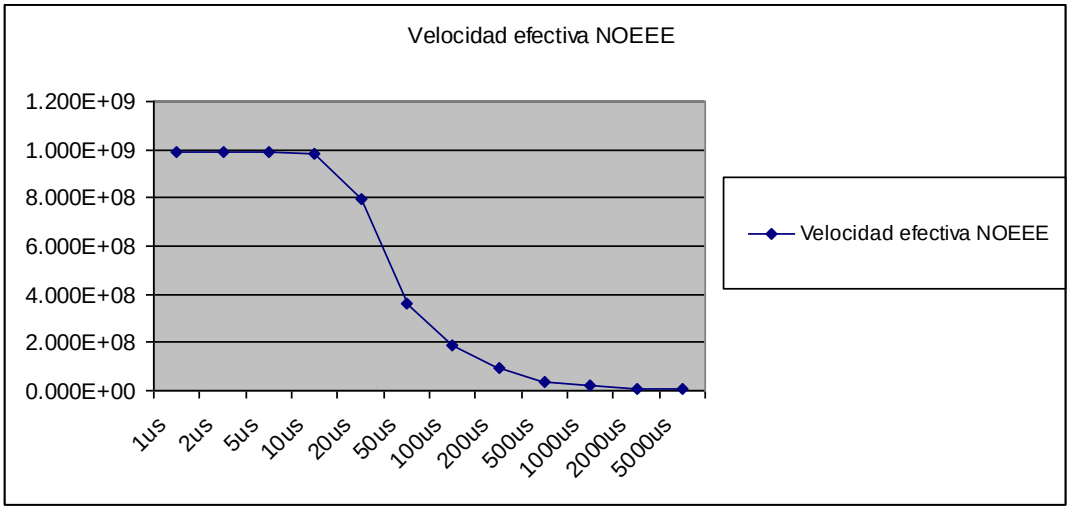

Fig. A2.40 Velocidad efectiva en función del delay de enlace

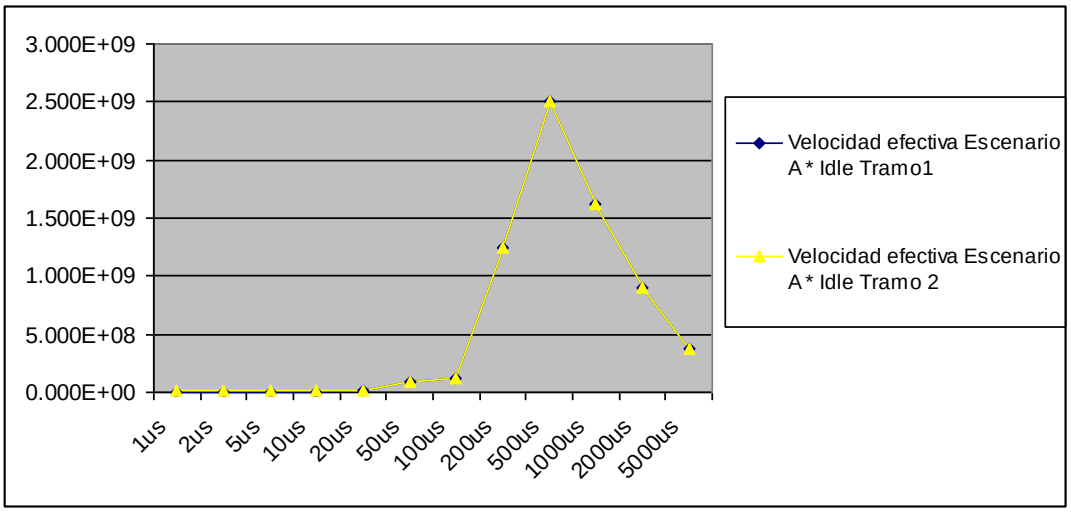

Fig. A2.41 Producto Idle por velocidad efectiva

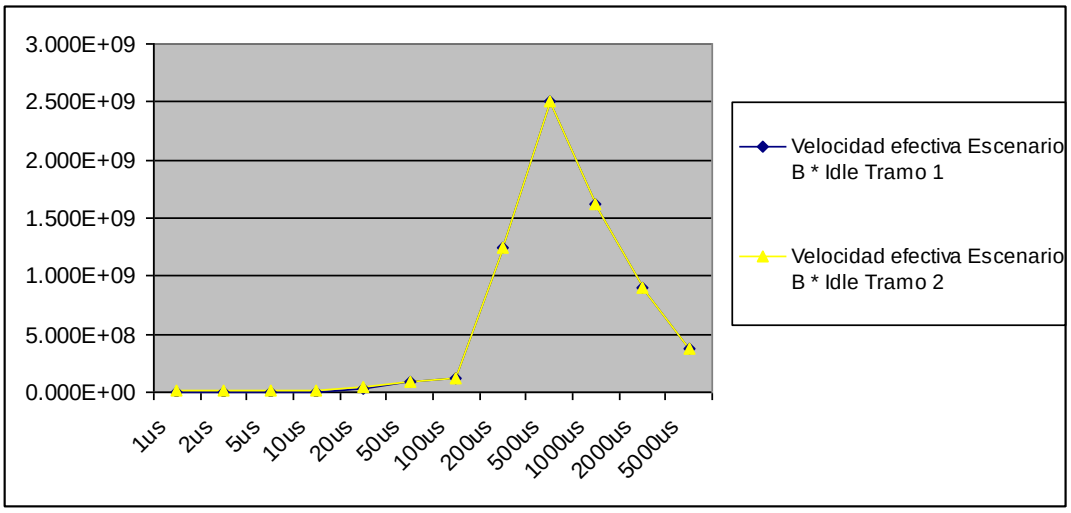

Fig. A2.42 Producto Idle por velocidad efectiva

5.2.4. Dos enlaces UDP

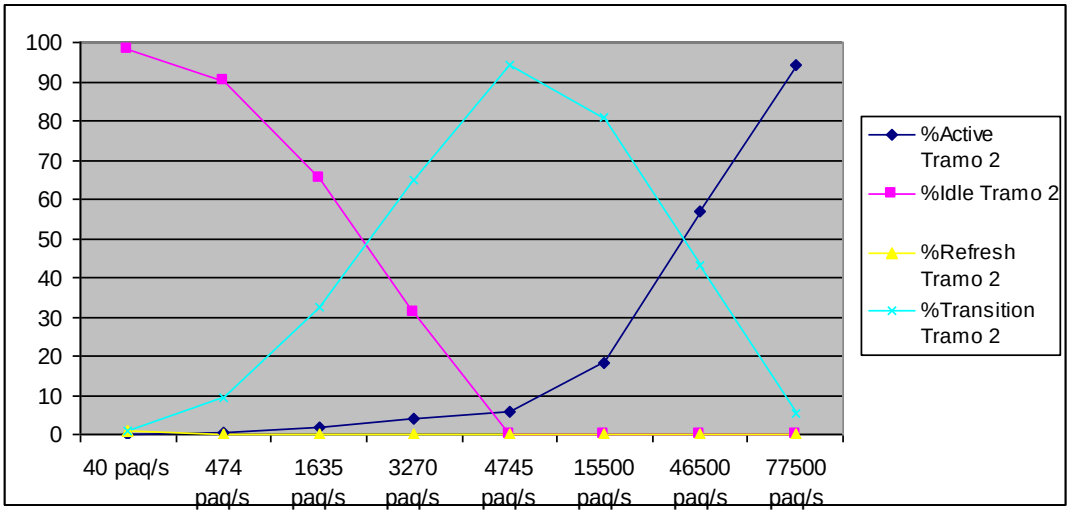

Fig. A2.43 % de tiempo en cada estado en función de la velocidad

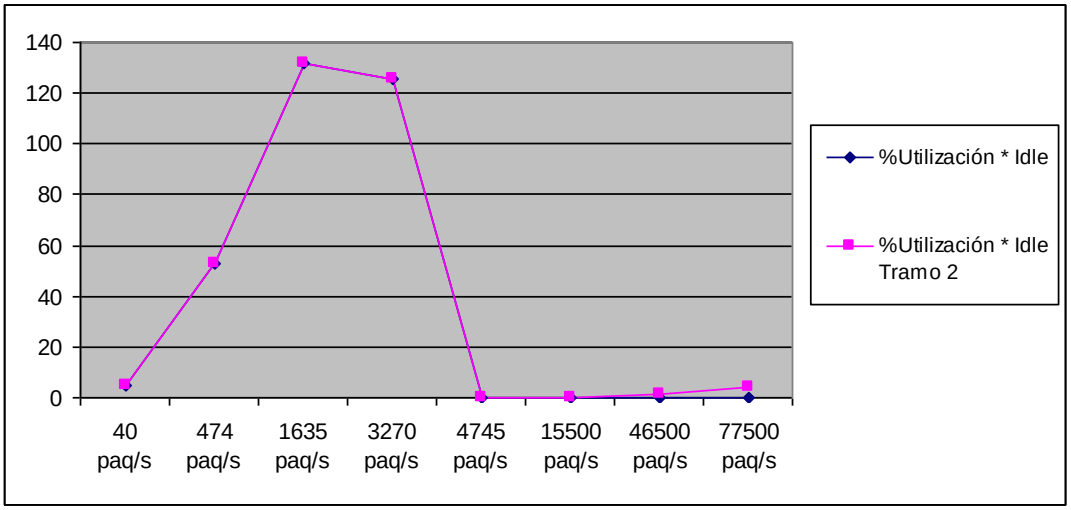

Fig. A2.44 Producto Idle por utilización

5.2.5. Tres enlaces UDP

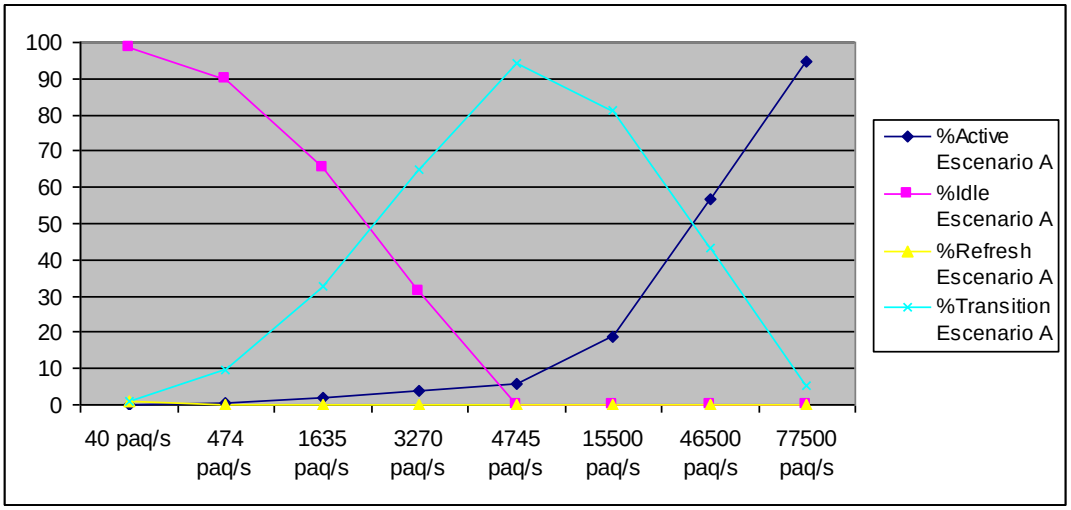

Fig. A2.45 de tiempo en cada estado en función de la velocidad

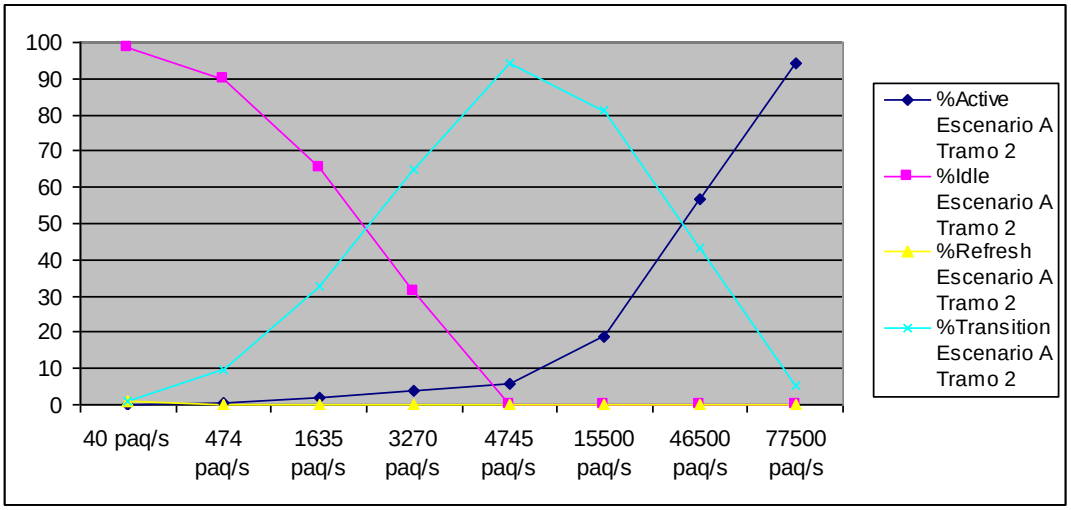

Fig. A2.46 de tiempo en cada estado en función de la velocidad

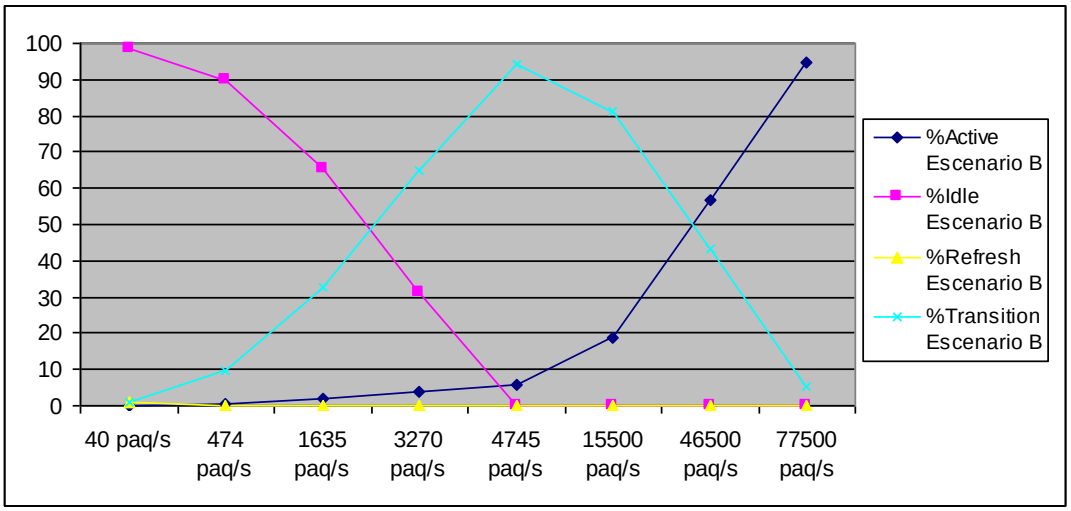

Fig. A2.47 de tiempo en cada estado en función de la velocidad

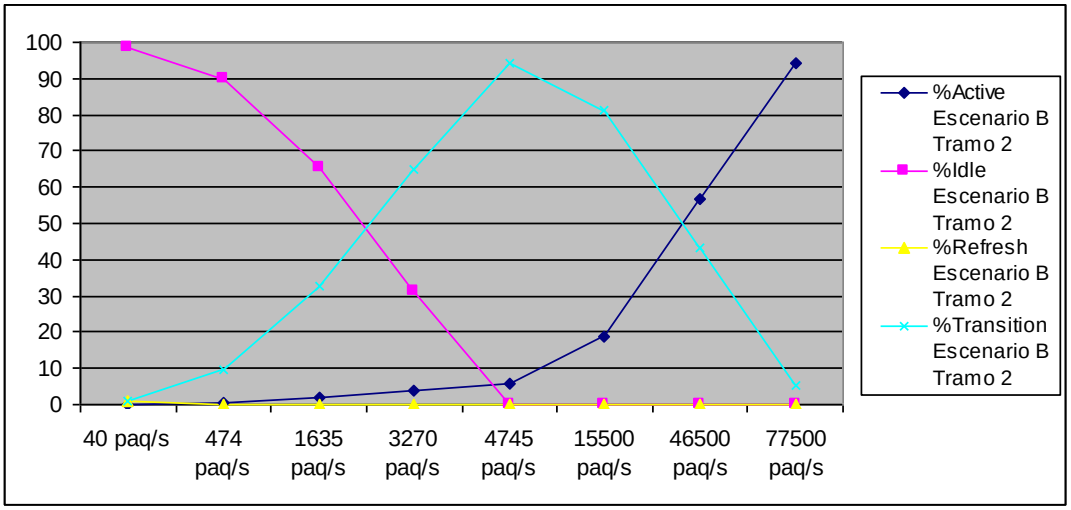

Fig. A2.48 de tiempo en cada estado en función de la velocidad

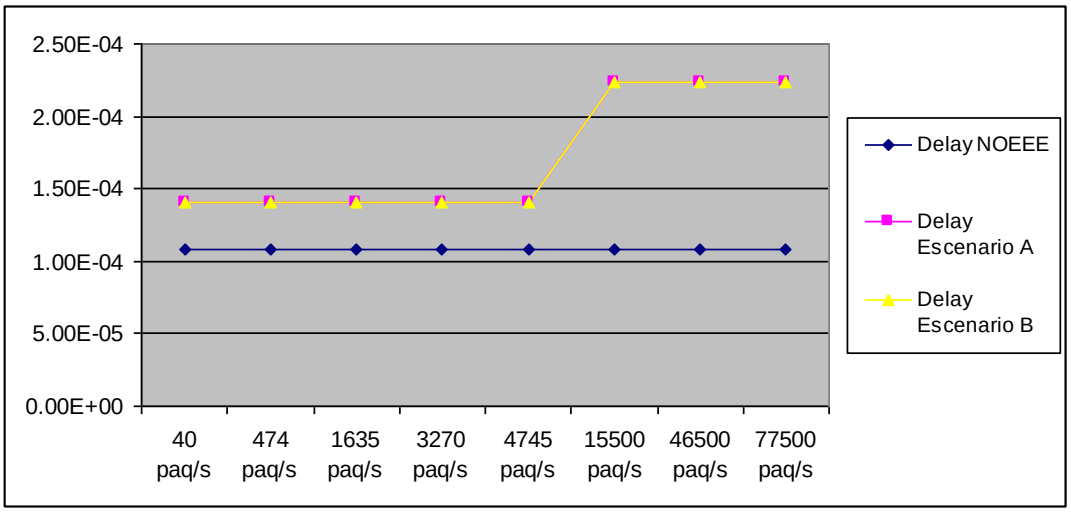

Fig. A2.49 Delay de extremo a extremo en función de la velocidad

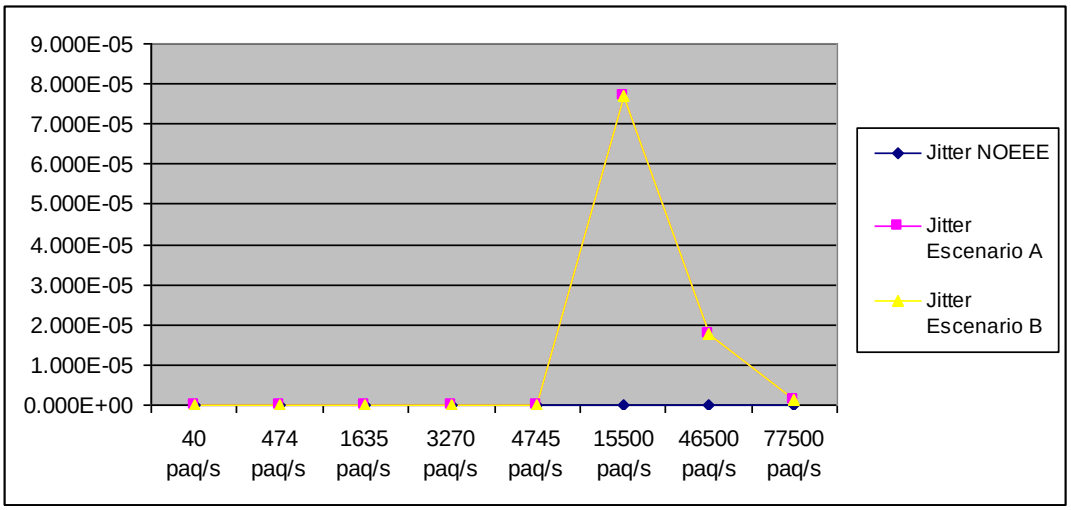

Fig. A2.50 Jitter en función de la velocidad

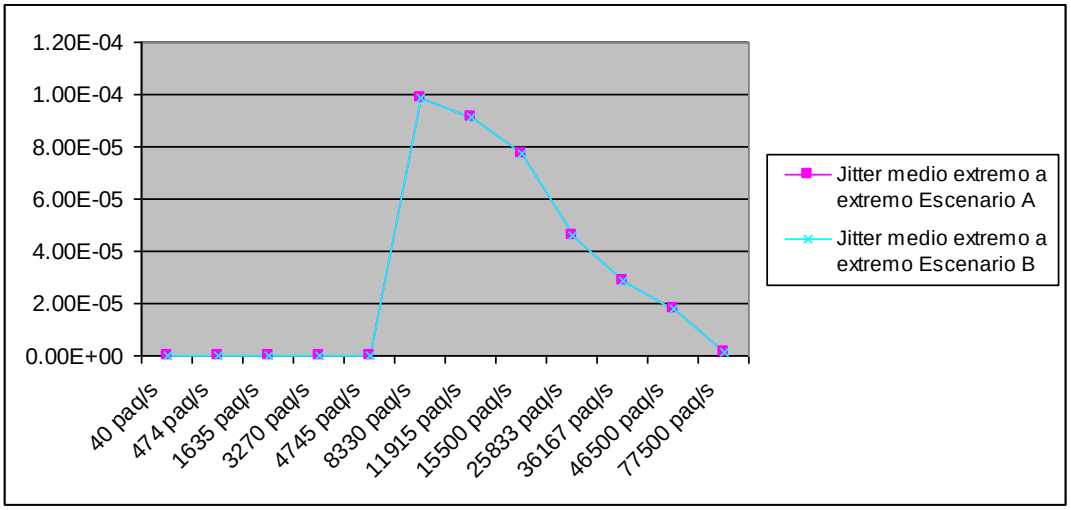

Fig. A2.51 Detalle del Jitter en función de la velocidad

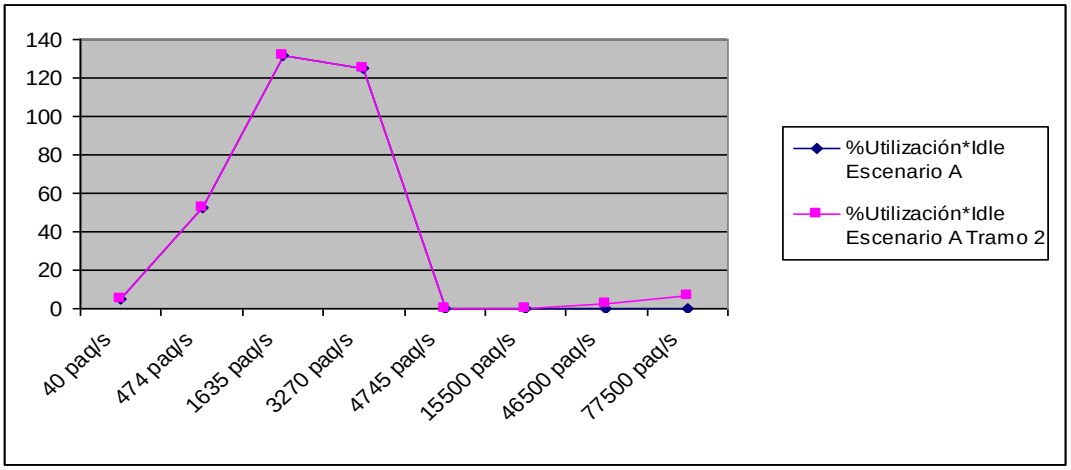

Fig. A2.52 Producto Idle por utilización

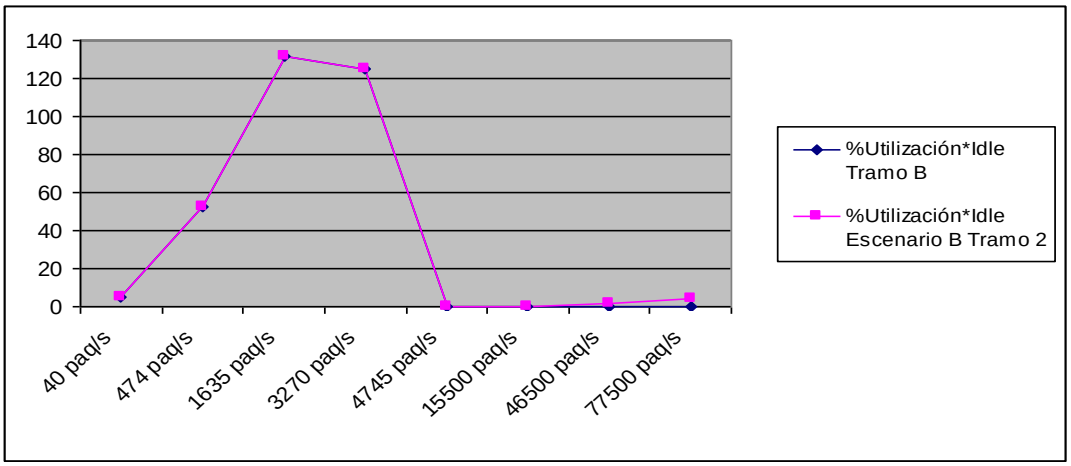

Fig. A2.53 Producto Idle por utilización

5.3. Anexo 3. Código OTcl de las simulaciones

5.3.1. Un enlace TCP

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#set nf [open out.nam w]
#$ns namtrace-all $nf


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]

lappend nodelist $n0
lappend nodelist $n1


set lan [$ns newLan $nodelist  1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]


set tcp [new Agent/TCP/Linux]
set sink [new Agent/TCPSink/Sack1]


$tcp set packetSize_ 1450
$tcp set timestamps_ true
$tcp set window_ 30000


$sink set ts_echo_rfc1323_ true


$ns attach-agent $n0 $tcp
$ns attach-agent $n1 $sink


$ns connect $tcp $sink


set ftp [new Application/FTP]
$ftp attach-agent $tcp


proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}

$ns at 0 "$tcp select_ca highspeed"
$ns at 0.00000001 "$ftp send 10000000"
$ns at 18.1 "finish"
```

```
$ns run
```

5.3.2. Dos enlaces TCP

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#set nf [open out.nam w]
#$ns namtrace-all $nf


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

lappend nodelist $n0
lappend nodelist $n1

lappend nodelist1 $n1
lappend nodelist1 $n2

set lan [$ns newLan $nodelist   1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist1  1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]

set tcp [new Agent/TCP/Linux]
set sink [new Agent/TCPSink/Sack1]


$tcp set packetSize_ 1450
$tcp set timestamps_ true
$tcp set window_ 30000


$sink set ts_echo_rfc1323_ true


$ns attach-agent $n0 $tcp
$ns attach-agent $n2 $sink


$ns connect $tcp $sink


set ftp [new Application/FTP]
$ftp attach-agent $tcp


proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
```

```
}

$ns at 0 "$tcp select_ca highspeed"
$ns at 0.00000001 "$ftp send 10000000"
$ns at 18.1 "finish"

$ns run
```

5.3.3. Tres enlaces TCP Escenario A

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#set nf [open out.nam w]
#$ns namtrace-all $nf


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

lappend nodelist $n0
lappend nodelist $n1

lappend nodelist1 $n1
lappend nodelist1 $n2

lappend nodelist2 $n2
lappend nodelist2 $n3

set lan [$ns newLan $nodelist   1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist1  1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredPhy]
set lan [$ns newLan $nodelist2  1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]


set tcp [new Agent/TCP/Linux]
set sink [new Agent/TCPSink/Sack1]


$tcp set packetSize_ 1450
$tcp set timestamps_ true
$tcp set window_ 30000


$sink set ts_echo_rfc1323_ true


$ns attach-agent $n0 $tcp
$ns attach-agent $n3 $sink
```

```
$ns connect $tcp $sink


set ftp [new Application/FTP]
$ftp attach-agent $tcp


proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}

$ns at 0 "$tcp select_ca highspeed"
$ns at 0.00000001 "$ftp send 10000000"
$ns at 18.1 "finish"


$ns run
```

5.3.4. Tres enlaces TCP Escenario B

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#set nf [open out.nam w]
#$ns namtrace-all $nf


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

lappend nodelist $n0
lappend nodelist $n1

lappend nodelist1 $n1
lappend nodelist1 $n2

lappend nodelist2 $n2
lappend nodelist2 $n3

set lan [$ns newLan $nodelist   1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist1  1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist2  1Gb  1us -macType Mac/802_3Full -
phyType Phy/WiredPhy]


set tcp [new Agent/TCP/Linux]
set sink [new Agent/TCPSink/Sack1]
```

```
$tcp set packetSize_ 1450
$tcp set timestamps_ true
$tcp set window_ 30000


$sink set ts_echo_rfc1323_ true


$ns attach-agent $n0 $tcp
$ns attach-agent $n3 $sink


$ns connect $tcp $sink


set ftp [new Application/FTP]
$ftp attach-agent $tcp


proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}

$ns at 0 "$tcp select_ca highspeed"
$ns at 0.00000001 "$ftp send 10000000"
$ns at 18.1 "finish"


$ns run
```

5.3.5. Un enlace UDP

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#$ns namtrace-all $namtrfd


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]



lappend nodelist $n0
lappend nodelist $n1


set lan [$ns newLan $nodelist  1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
```

```
#Setup a mUDP connection

set udp0 [new Agent/mUDP]

#set the sender trace file name (sd)

$udp0 set_filename sd
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.025

$cbr0 attach-agent $udp0

set null0 [new Agent/mUdpSink]

#set the receiver trace file name (rd)

$null0 set_filename rd
$ns attach-agent $n1 $null0


$ns connect $udp0 $null0



proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}

$ns at 0.000182 "$cbr0 start"

$ns at 250.000182 "finish"

$ns run
```

5.3.6. Dos enlaces UDP

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#$ns namtrace-all $namtrfd


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]


lappend nodelist $n0
lappend nodelist $n1
```

```
lappend nodelist1 $n1
lappend nodelist1 $n2



set lan [$ns newLan $nodelist  1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist1 1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]

#Setup a mUDP connection

set udp0 [new Agent/mUDP]

#set the sender trace file name (sd)

$udp0 set_filename sd
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.025

$cbr0 attach-agent $udp0

set null0 [new Agent/mUdpSink]

#set the receiver trace file name (rd)

$null0 set_filename rd
$ns attach-agent $n2 $null0


$ns connect $udp0 $null0



proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}

$ns at 0.000182 "$cbr0 start"

$ns at 250.000182 "finish"

$ns run
```

5.3.7. Tres enlaces UDP Escenario A

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
```

```
#$ns namtrace-all $namtrfd


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

lappend nodelist $n0
lappend nodelist $n1

lappend nodelist1 $n1
lappend nodelist1 $n2

lappend nodelist2 $n2
lappend nodelist2 $n3


set lan [$ns newLan $nodelist  1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist1 1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredPhy]
set lan [$ns newLan $nodelist2 1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]

#Setup a mUDP connection

set udp0 [new Agent/mUDP]

#set the sender trace file name (sd)

$udp0 set_filename sd
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.025

$cbr0 attach-agent $udp0

set null0 [new Agent/mUdpSink]

#set the receiver trace file name (rd)

$null0 set_filename rd
$ns attach-agent $n3 $null0


$ns connect $udp0 $null0



proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}
```

```
$ns at 0.000182 "$cbr0 start"

$ns at 250.000182 "finish"

$ns run
```

5.3.8. Tres enlaces UDP Escenario B

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#$ns namtrace-all $namtrfd


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

lappend nodelist $n0
lappend nodelist $n1

lappend nodelist1 $n1
lappend nodelist1 $n2

lappend nodelist2 $n2
lappend nodelist2 $n3


set lan [$ns newLan $nodelist  1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist1 1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]
set lan [$ns newLan $nodelist2 1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredPhy]

#Setup a mUDP connection

set udp0 [new Agent/mUDP]

#set the sender trace file name (sd)

$udp0 set_filename sd
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.025

$cbr0 attach-agent $udp0

set null0 [new Agent/mUdpSink]

#set the receiver trace file name (rd)
```

```
$null0 set_filename rd
$ns attach-agent $n3 $null0


$ns connect $udp0 $null0



proc finish {} {
  global ns trfd
  $ns flush-trace
  close $trfd
  exit 0
}

$ns at 0.000180 "$cbr0 start"

$ns at 250.000180 "finish"

$ns run
```

5.3.9. Un enlace con fuente de vídeo

```
set ns [new Simulator]
set trfd [open nsgatewayLinux.tr w]
#set namtrfd [open namnsgatewayLinux.tr w]
$ns trace-all $trfd
#$ns namtrace-all $namtrfd


$ns node-config -macType Mac/802_3Full -phyType Phy/WiredEEEPhy

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]



lappend nodelist1 $n1
lappend nodelist1 $n2


$ns duplex-link $n0 $n1 1Gb 1us DropTail
$ns queue-limit $n0 $n1 10000000000


set lan [$ns newLan $nodelist1 1Gb 10us -macType Mac/802_3Full -
phyType Phy/WiredEEEPhy]


#Setup a mUDP connection

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0


set vdo [new Application/Traffic/MPEG4]
$vdo set initialSeed_ 0.4
```

```
$vdo set rateFactor_ 5
$vdo attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n2 $null0

$ns connect $udp0 $null0

proc finish {} {
global ns trfd
$ns flush-trace
close $trfd
exit 0
}

$ns at 0.000180 "$vdo start"
$ns at 2.5 "$vdo stop"
$ns at 2.5 "finish"
$ns run
```