

MODELO ALGORÍTMICO PARA LIMITAR LA CONGESTIÓN DEL PROTOCOLO TCP EN REDES DE EXTREMO A EXTREMO

Autor

Andrés Felipe Hernández León

Tutor

Octavio Salcedo Parra

PhD En Ingeniería Informática

PhD En Estudios Políticos

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Maestría en Ciencias de la Información y las Comunicaciones

Énfasis en Teleinformática

Bogotá, Colombia

junio de 2017

TABLA DE CONTENIDO

RESUMEN 6

ABSTRACT 7

PALABRAS CLAVE 7

LISTA DE FIGURAS 8

LISTA DE TABLAS 11

LISTA DE ECUACIONES 14

INTRODUCCIÓN 16

1. PROBLEMA DE INVESTIGACIÓN 17

1.1 PLANTEAMIENTO DEL PROBLEMA 17

1.2 OBJETIVOS 19

1.2.1 OBJETIVO GENERAL 19

1.2.2 OBJETIVOS ESPECÍFICOS 19

1.3 RESULTADOS ESPERADOS 19

2. MARCO DE REFERENCIA 21

2.1 MARCO CONCEPTUAL 21

2.1.1 REDES DE EXTREMO A EXTREMO 21

2.1.1.1 DEFINICIÓN 21

2.1.1.2 ARQUITECTURA 21

2.1.1.3 APLICACIONES 24

2.1.2 TCP 26

2.1.3 VENTANA DESLIZANTE 28

2.1.4 ACK (ACKNOWLEDGEMENT) 29
2.1.5 CONGESTIÓN DE RED 29
2.1.6 ALGORITMO DE CONTROL DE CONGESTIÓN 30
2.2 MARCO TEÓRICO 30
2.2.1 CONGESTIÓN EN DIFERENTES ETAPAS DEL INTERNET 30
2.2.2 INVESTIGACIONES PREVIAS CON ALGORITMO TCP RENO Y TCP VEGAS 35
2.2.3 INVESTIGACIONES PREVIAS CON ALGORITMO TCP NEWRENO 35
2.2.4 INVESTIGACIONES PREVIAS DE CONGESTIÓN EN REDES EXTREMO A EXTREMO 36
2.2.5 INVESTIGACIONES PREVIAS DE MÉTRICAS DE CONTROL DE CONGESTIÓN 37
2.3 ESTADO DEL ARTE 39
2.3.1 PRUEBAS DE ESTRÉS 39
2.3.2 METODOLOGÍA 40
2.3.3 MODELOS A DISCUTIR 41
2.3.4 DISCUSIÓN DE RESULTADOS 44
2.3.5 COMPARACIÓN DE RESULTADOS CON OTROS AUTORES 63
2.3.6 CONCLUSIONES ESTADO DEL ARTE 65
3. DISEÑO DEL MODELO ALGORÍTMICO PARA LIMITAR LA CONGESTIÓN DEL PROTOCOLO TCP EN REDES DE EXTREMO A EXTREMO 67
3.1 BUFFER 67
3.2 TAMAÑO DE LOS PAQUETES 70
3.3 RELACIÓN ENTRE TAMAÑO DEL BUFFER Y EL TAMAÑO DE LOS PAQUETES 72

3.4 TIEMPO DE LOS ACK 73
3.5 TIPOS DE SERVICIOS 76
3.6 RELACIÓN ENTRE TIEMPOS DE ACK Y LOS SERVICIOS A PRESTAR 77
3.7 MODELO ALGORITMO PARA LIMITAR LA CONGESTIÓN DEL PROTOCOLO TCP EN REDES DE EXTREMO A EXTREMO 79
4. HERRAMIENTAS DE SIMULACIÓN 81
4.1 NETWORK SIMULATOR 3 (NS3) 81
4.2 QUALNET 81
4.3 JiST 82
4.4 OMNET++ 83
4.5 NETWORK SIMULATOR 2 (NS2) 84
5. DISEÑO E IMPLEMENTACION DE PRUEBAS 86
5.1 DISEÑO DE PROTOCOLO DE PRUEBAS 86
5.2 OBTENCIÓN DE PARÁMETROS REALES 87
5.3 ANÁLISIS DE LAS MÉTRICAS OBTENIDAS 89
5.4 SIMULACIÓN DE LA RED SEGÚN LAS MEDICIONES OBTENIDAS 92
5.5 SIMULACIÓN DEL TOTAL DE ESCENARIOS 94
5.6 ANÁLISIS DE RESULTADOS 107
5.7 COMPARACIÓN CON TRABAJOS RELACIONADOS 117
6. CONCLUSIONES, TRABAJOS FUTUROS Y APORTES DE LA INVESTIGACION 124
6.1 CONCLUSIONES 124
6.2 TRABAJOS FUTUROS 127
6.3 APORTES DE LA INVESTIGACIÓN 128
7. REFERENCIAS 129

ANEXOS..... 136

RESUMEN

En este trabajo de investigación se presenta un modelo algorítmico para limitar la congestión en redes TCP de extremo a extremo, para ello fue realizada una investigación del estado del arte de este tipo de redes en la actualidad y con ello se logró determinar las falencias que estas presentan en la actualidad.

Fue realizada una investigación acerca de las métricas que son más importantes y determinantes al momento de las trasmisiones TCP de extremo a extremo y como éstas deben ser monitoreadas y configuradas para así obtener el mejor resultado bajo ciertas condiciones previamente identificadas de la red.

Posterior a esto se procedió a diseñar y presentar el modelo como una serie de pasos a seguir según los factores reales que la red de extremo a extremo presenta, seguido de él diseño indicado igualmente a manera de pasos de una metodología de pruebas del modelo sobre simulaciones usando el software ns2.

Ya listas las simulaciones estas se presentan para la totalidad de los escenarios encontrados en comparación con los resultados de una red real de extremo a extremo con el fin de determinar la efectividad del modelo diseñado.

Finalmente se realizó una serie de recomendaciones y se enumeran las conclusiones que la investigación arrojó así como una serie de trabajos futuros.

ABSTRACT

An algorithmic model is presented to limit congestion in end-to-end TCP networks, a state-of-the-art investigation of current networks of this type is carried out and the faults present are determined.

This work have an investigation about the metrics that are most important and determinant at the moment of the end-to-end TCP transmissions and how they must be monitored and configured in order to obtain the best result under certain previously identified conditions of the network.

The model is designed and presented as a series of steps to follow according to the actual factors that the end-to-end network presents, followed by its design also indicated as steps of a model testing methodology on simulations using ns2 software.

Simulations of all the scenarios encountered are presented in comparison to the results of an actual end-to-end network in order to determine the effectiveness of the designed model.

A series of recommendations are made and the conclusions drawn by the research are listed as well as a series of future work.

PALABRAS CLAVE

Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Extremo a Extremo, congestión, interconexión de redes protocolo, buffer, acknowledgement (ACK), , ancho de banda, simulación, métricas de red, calidad de servicio.

LISTA DE FIGURAS

Figura 1 Arquitectura Centralizada Redes Extremo a Extremo. 22
Figura 2 Arquitectura Descentralizada Redes Extremo a Extremo 23
Figura 3 Arquitectura Mixta Redes Extremo a Extremo 24
Figura 4 Esquema de red extremo a extremo a probar 41
Figura 5 Esquema de red extremo a extremo para la prueba con google.com 42
Figura 6 Esquema de red extremo a extremo para la prueba con youtube.com 42
Figura 7 Esquema de red extremo a extremo para la prueba con eltiempo.com 43
Figura 8 Esquema de red extremo a extremo para la prueba con caracoltv.com 43
Figura 9 Esquema de red extremo a extremo para la prueba con facebook.com 44
Figura 10 Gráfica de resultados pruebas de estrés servidor google.com día 1 49
Figura 11 Gráfica de resultados pruebas de estrés servidor google.com día 2 50
Figura 12 Gráfica de resultados pruebas de estrés servidor google.com día 3 50
Figura 13 Gráfica de resultados pruebas de estrés servidor google.com día 4 51
Figura 14 Gráfica de resultados pruebas de estrés servidor google.com día 5 51
Figura 15 Gráfica de resultados pruebas de estrés servidor youtube.com día 1 52
Figura 16 Gráfica de resultados pruebas de estrés servidor youtube.com día 2 52
Figura 17 Gráfica de resultados pruebas de estrés servidor youtube.com día 3 53
Figura 18 Gráfica de resultados pruebas de estrés servidor youtube.com día 4 53
Figura 19 Gráfica de resultados pruebas de estrés servidor youtube.com día 5 54
Figura 20 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 1 54
Figura 21 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 2 55
Figura 22 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 3 55
Figura 23 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 4 56
Figura 24 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 5 56
Figura 25 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 1 57
Figura 26 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 2 57
Figura 27 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 3 58

Figura 28 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 4 58
Figura 29 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 5 59
Figura 30 Gráfica de resultados pruebas de estrés servidor facebook.com día 1 59
Figura 31 Gráfica de resultados pruebas de estrés servidor facebook.com día 2 60
Figura 32 Gráfica de resultados pruebas de estrés servidor facebook.com día 3 60
Figura 33 Gráfica de resultados pruebas de estrés servidor facebook.com día 4 61
Figura 34 Gráfica de resultados pruebas de estrés servidor facebook.com día 5 61
Figura 35 Ejemplo de la diferencia de capacidades de buffers en routers de una misma LAN 68
Figura 36 Modelo M/M/1/K 68
Figura 37 Modelo algorítmico para limitar la congestión del protocolo TCP en redes de extremo a extremo 80
Figura 38 Estructura en diagrama de bloques simulador ns3 81
Figura 39 Estructura en diagrama de bloques simulador QUALNET 82
Figura 40 Estructura en diagrama de bloques simulador JiST 83
Figura 41 Ejemplo de simulación en OMNET++ 84
Figura 42 Estructura en diagrama de bloques simulador ns2 85
Figura 43 Modelo de red LAN-WAN-LAN a simular 92
Figura 44 Componentes de la red LAN-WAN-LAN a simular 93
Figura 45 Simulación de la red real en ns2 94
Figura 46 Relación entre paquetes del tamaño máximo (64KB) y de 50KB 108
Figura 47 Relación entre paquetes del tamaño máximo (64KB) y de 40KB 109
Figura 48 Relación entre paquetes del tamaño máximo (64KB) y de 30KB 110
Figura 49 Relación entre paquetes del tamaño máximo (64KB) y de 20KB 111
Figura 50 Relación entre paquetes del tamaño máximo (64KB) y de 10KB 112
Figura 51 Relación entre paquetes del tamaño máximo (64KB) y de 1KB 113
Figura 52 Cantidad de error con ancho de banda (1.15Mb) variando tamaño del paquete 114
Figura 53 Comparación de la eficiencia del modelo con respecto a los datos reales y a la simulación de la red real 117

Figura 54 Resultados del control de congestión del paper Unleashing Tor, BitTorrent & Co How to Relieve TCP Deficiencies in Overlays 118
Figura 55 Topología de red sobre la que se probo el mecanismo del control de congestión 119
Figura 56 Topología de red sobre la que se probó el mecanismo del control de congestión 120
Figura 57 Topología de red sobre la que se probó el mecanismo del control de congestión 121
Figura 58 Topología de red sobre la que se probó el mecanismo del control de congestión 122

LISTA DE TABLAS

Tabla 1 Datos de los extremos pruebas google.com 44
Tabla 2 Resultados pruebas de estrés servidor google.com 45
Tabla 3 Datos de los extremos pruebas youtube.com 45
Tabla 4 Resultados pruebas de estrés servidor youtube.com 46
Tabla 5 Datos de los extremos pruebas eltiempo.com 46
Tabla 6 Resultados pruebas de estrés servidor eltiempo.com 47
Tabla 7 Datos de los extremos pruebas caracoltv.com 47
Tabla 8 Resultados pruebas de estrés servidor caracoltv.com 48
Tabla 9 Datos de los extremos pruebas facebook.com 48
Tabla 10 Resultados pruebas de estrés servidor facebook.com 49
Tabla 11 Resultados Simulaciones en 4 escenarios 64
Tabla 12 Capacidades de buffers definidas y su correspondiente clasificación 70
Tabla 13 Tamaños de los paquetes a trasmitir y su correspondiente clasificación 72
Tabla 14 Relación entre la cantidad de buffer disponible y el tamaño de los paquetes 73
Tabla 15 Porcentajes promedio de seguimientos de diversos servicios en redes LAN-WAN-LAN 77
Tabla 16 Relación entre los servicios a habilitar y el tiempo de los ACK para la red LAN 78
Tabla 17 Relación entre los servicios a habilitar y el tiempo de los ACK para la red WAN78
Tabla 18 Porcentaje de error de los 5 días de seguimiento en la red LAN-WAN-LAN entre Bogotá y Medellín 88
Tabla 19 Resumen de las métricas obtenidas de los 5 días de seguimiento de la red LAN-WAN-LAN entre Bogotá y Medellín los días 27 al 31 de Marzo 89
Tabla 20 Parámetros de la red real usados para la simulación de la misma 91
Tabla 21 Resultados obtenidos de la simulación de la red real, fuente creación propia ... 94
Tabla 22 Rango de valores obtenidos para las métricas calculados siguiendo las instrucciones del modelo, fuente creación propia. 95
Tabla 23 Valores de las métricas escenario 1 95

Tabla 24 Valores de las métricas escenario 2..... 96
Tabla 25 Valores de las métricas escenario 3..... 96
Tabla 26 Valores de las métricas escenario 4..... 96
Tabla 27 Valores de las métricas escenario 5..... 97
Tabla 28 Valores de las métricas escenario 6..... 97
Tabla 29 Valores de las métricas escenario 7..... 97
Tabla 30 Valores de las métricas escenario 8..... 98
Tabla 31 Valores de las métricas escenario 9..... 98
Tabla 32 Valores de las métricas escenario 10..... 98
Tabla 33 Valores de las métricas escenario 11..... 99
Tabla 34 Valores de las métricas escenario 12..... 99
Tabla 35 Valores de las métricas escenario 13..... 99
Tabla 36 Valores de las métricas escenario 14..... 100
Tabla 37 Valores de las métricas escenario 15..... 100
Tabla 38 Valores de las métricas escenario 16..... 100
Tabla 39 Valores de las métricas escenario 17..... 101
Tabla 40 Valores de las métricas escenario 18..... 101
Tabla 41 Valores de las métricas escenario 19..... 101
Tabla 42 Valores de las métricas escenario 20..... 102
Tabla 43 Valores de las métricas escenario 21..... 102
Tabla 44 Valores de las métricas escenario 22..... 102
Tabla 45 Valores de las métricas escenario 23..... 103
Tabla 46 Valores de las métricas escenario 24..... 103
Tabla 47 Valores de las métricas escenario 25..... 103
Tabla 48 Valores de las métricas escenario 26..... 104
Tabla 49 Valores de las métricas escenario 27..... 104
Tabla 50 Valores de las métricas escenario 28..... 104
Tabla 51 Valores de las métricas escenario 29..... 105
Tabla 52 Valores de las métricas escenario 30..... 105
Tabla 53 Valores de las métricas escenario 31..... 105

Tabla 54 Valores de las métricas escenario 32 106
Tabla 55 Valores de las métricas escenario 33 106
Tabla 56 Valores de las métricas escenario 34 106
Tabla 57 Valores de las métricas escenario 35 107
Tabla 58 Valores de las métricas escenario 36 107
Tabla 59 Comparación del porcentaje de mejora del modelo con respecto a la red real en cada uno de los escenarios 116
Tabla 60 Comparación de las mejoras del control de congestión del presente trabajo con otros realizados con anterioridad por otros autores 123

LISTA DE ECUACIONES

Ecuación 1 Intensidad del sistema en un modelo de colas M/M/1/K 69
Ecuación 2 Intensidad del sistema en una cola M/M/1/K sin congestión 69
Ecuación 3 Intensidad del sistema en una cola M/M/1/K estable 69
Ecuación 4 Intensidad del sistema en una cola M/M/1/K cuando se presenta congestión 69
Ecuación 5 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 1 Bogotá-Medellín en milisegundos 74
Ecuación 6 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 2 Bogotá-Medellín en milisegundos 74
Ecuación 7 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 3 Bogotá-Medellín en milisegundos 74
Ecuación 8 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 4 Bogotá-Medellín en milisegundos 74
Ecuación 9 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 5 Bogotá-Medellín en milisegundos 74
Ecuación 10 Gran media de los tiempos de los ACK de la red LAN..... 75
Ecuación 11 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 1 Bogotá-Medellín en milisegundos 75
Ecuación 12 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 2 Bogotá-Medellín en milisegundos 75
Ecuación 13 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 3 Bogotá-Medellín en milisegundos 75
Ecuación 14 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 4 Bogotá-Medellín en milisegundos 75
Ecuación 15 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 5 Bogotá-Medellín en milisegundos 75
Ecuación 16 Gran media de los tiempos de los ACK de la red WAN 76
Ecuación 17 Tamaño el paquete de la red LAN y WAN entre Bogotá y Medellín..... 89

Ecuación 18 Cantidad promedio de usuarios concurrentes red LAN 89
Ecuación 19 Cantidad promedio de usuarios concurrentes red WAN 89
Ecuación 20 Cálculo de la cantidad de paquetes que caben en el buffer por usuario red LAN 90
Ecuación 21 Cálculo de la cantidad de paquetes que caben en el buffer para enlace WAN 90
Ecuación 22 Cantidad promedio de tiempo de los ACK red LAN 90
Ecuación 23 Cantidad promedio de tiempo de los ACK red WAN 90
Ecuación 24 Ancho de Banda promedio para cada usuario red LAN 90
Ecuación 25 Cálculo de ancho de banda de la red LAN con los 6 servicios TCP en congestión 91
Ecuación 26 Ancho de banda promedio para cada enlace red WAN 91
Ecuación 27 Cálculo de ancho de banda de la red WAN con los 6 servicios TCP en congestión 91

INTRODUCCIÓN

Desde que internet dejó de ser algo exclusivamente militar y/o académico y empezó a tener cada vez más un uso comercial, a la vez que cada día empezó a ser más fácil su acceso (Kurose & Ross , 2004) La cantidad de usuarios y de tráfico empezó a crecer constantemente lo cual con el paso de los años ha generado congestión, esto hace que las redes sean ineficientes, puedan presentar errores y demás aspectos que disminuyen una buena calidad del servicio.

Es por esto que surge la necesidad de crear algoritmos del control de congestión, para el caso específico de este trabajo: un algoritmo para redes de extremo a extremo, iniciando con la identificación de las causas actuales de la congestión con el fin de poder “atacar” el problema directamente, seguido de un análisis de los actuales algoritmos el cual se enfoca en el por qué estos han disminuido su eficiencia en la actualidad, una vez se dispone de esta información se realiza el modelo de algoritmo para finalmente probarlo en varios escenarios mediante simulaciones y terminar con un análisis de los resultados los cuales se espera sean satisfactorios reduciendo la congestión que la red puede presentar, todo esto enfocado en redes de tipo extremo a extremo.

1. PROBLEMA DE INVESTIGACIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Desde sus inicios hasta la actualidad las redes de extremo a extremo ya sean cableadas, inalámbricas o satelitales (Obata, Nishimoto, & Ishida, 2009) han presentado un crecimiento cada vez más acelerado tanto en sus niveles de tráfico como en la cantidad de usuarios que hacen uso de estas (Alrshah, Othman, Ali, & Hanapi, 2015) ya sean los usuarios convencionales es decir ubicados cronológicamente en los inicios de internet se pensaba sólo serían conexiones desde computadores o los procedentes de la convergencia de internet con otras tecnologías como la telefonía celular tanto en voz como en datos las trasmisiones en vivo de emisoras de radio las llamadas y video llamadas telefónicas y el streaming de canales de televisión.

Dichas integraciones a diario representan millones de usuarios concurrentes que generan niveles de tráfico sumamente altos (Luoa, Ci, Wu, & Tang, 2010), consecuencia de lo anterior se ha observado que cada vez es mayor la carga de trabajo para el protocolo TCP, que aunque tenga actualmente algoritmos del control de congestión por lo ya mencionado está recibiendo un nivel de transmisiones en algunos casos mayor a las que realmente puede procesar de manera correcta presentando retardos (Rath & Karandikar, 2011), paquetes perdidos, la necesidad de retrasmisiones por paquetes dañados o incompletos, entre otros aspectos que afectan claramente la calidad del servicio.

Una solución para esto es el algoritmo de control de congestión TCP Tahoe que principalmente busca solucionar los primeros problemas que se encontraron cuando la implementación de internet inició formalmente incluyendo nuevos términos como: inicio lento (Slow-Start) y Retrasmisión Rápida (Fast Retransmit) (Practicas de Redes, Protocolo TCP, 2003). Su principal inconveniente consiste en que cuando el algoritmo detecta que los paquetes se perdían (por vencimiento de tiempos) reducía el tamaño de la ventana a su

valor mínimo lo cual claramente significaba una reducción en los tiempos de recuperación (Salazar, 2007).

Otra solución es el algoritmo de control de congestión TCP RENO el cual en principio cumple con las características de su antecesor el TCP Tahoe incluyendo un algoritmo de recuperación rápida el cual trabaja paralelamente con el algoritmo de retrasmisión rápida (Jitendra Padhye, 2000). Aun así el principal inconveniente de TCP RENO consiste en que cuando se ve obligado a reducir la ventana de congestión lo hace múltiples veces (Bogdan Moraru, 2001).

Una tercera solución es el algoritmo TCP NEW RENO, el cual como se puede deducir de su nombre es una modificación del algoritmo TCP RENO, su debilidad consiste en que NEW RENO no sale del estado de retransmisión rápida hasta que recibe ACK para todos los datos que estaban pendientes en el momento en que entró a el estado ya mencionado, también cabe mencionar que TCP NEW RENO presenta un Round-Trip Delay Time independiente para detectar cada pérdida de paquetes, además de que solo puede deducir una pérdida de paquetes una vez ha recibido un ACK por el primer segmento retransmitido (Practicas de Redes, Protocolo TCP, 2003).

Una cuarta solución es el algoritmo de control de congestión TCP CUBIC, este algoritmo como principal característica mantiene el sincronismo de los ACK para poder incrementar el tamaño de su ventana además que cuando el algoritmo entra en la fase de recuperación rápida no realiza cambios en ninguna de sus etapas, su principal debilidad consiste en que el control de la ventana es demasiado complejo para redes pequeñas además de resultar bastante perjudicial para redes de baja velocidad.

De todo lo mencionado se puede deducir que si la situación persiste y como ya se mencionó las redes de internet siguen creciendo al igual que la cantidad de usuarios y sus respectivas transmisiones si no se implementa algún correctivo con respecto a la congestión se llegará al punto en que las redes de extremo a extremo actuales pueden llegar a tener niveles bajos de transmisiones satisfactorias y gran lentitud, además de posibles colapsos a nivel de infraestructura y disminución evidente y notable de la calidad del servicio.

Es por esto que es necesario desarrollar un modelo algorítmico que logre limitar la congestión del protocolo TCP en redes de extremo a extremo el cual logrará reducir los retardos, trasmisiones fallidas y problemas de conexión facilitando en un futuro el crecimiento continuo de redes de extremo a extremo con el actual protocolo TCP.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Crear un modelo de algoritmo que permita reducir el problema de congestión del actual protocolo TCP en redes de extremo a extremo.

1.2.2 OBJETIVOS ESPECÍFICOS

- Analizar las métricas de algoritmos existentes y realizar pruebas de sistemas con el fin de observar el estado actual del funcionamiento de los mismos.

- Plantear una combinación de métricas y parámetros de diferentes algoritmos (Tamaño del segmento, Tamaño del Buffer, Velocidad de transferencia, Tiempos de los ACK) para lograr una mejor administración de la congestión de TCP.

- Realizar el modelo de algoritmo de control de congestión que se base en los resultados de la investigación.

- Desarrollar e implementar un protocolo de pruebas que permita validar, el modelo realizado, con el fin de verificar la eficiencia de los resultados.

1.3 RESULTADOS ESPERADOS

En cuanto al impacto el proyecto lo presentará a nivel local, nacional e internacional, ya que se pretende hacer una contribución a una mejor eficiencia del protocolo TCP, además del

fortalecimiento de la capacidad investigativa del grupo de investigación Internet Inteligente de la Universidad Distrital Francisco José de Caldas.

En cuanto a los resultados se tienen:

- Un documento de revisión del estado del arte que evidencia los principales inconvenientes que en la actualidad experimenta el protocolo TCP en redes de extremo a extremo.

- Un modelo algorítmico que reducirá la congestión del protocolo TCP evitando retrasos, retrasmisiones, trasmisiones fallidas, errores en aplicaciones.

- Un protocolo de pruebas que permita una correcta implementación, puesta en operación y recolección de resultados para las simulaciones de este modelo algorítmico que garanticen la calidad de los mismos.

2. MARCO DE REFERENCIA

2.1 MARCO CONCEPTUAL

2.1.1 REDES DE EXTREMO A EXTREMO

2.1.1.1 DEFINICIÓN

Una red de extremo a extremo es una red que no tiene ni clientes ni servidores en un rol fijo, por el contrario está conformada por una serie de equipos que se pueden comportar a la vez como clientes y como servidores con respecto a los demás equipos de la red, se podría decir que es el equivalente opuesto al modelo cliente-servidor que ha existido desde los inicios de internet. En este tipo de redes todos los equipos se comportan igual y pueden realizar el mismo tipo de operaciones y soportar las mismas aplicaciones, aun así se diferencian en aspectos como su configuración, su velocidad de procesamiento, cantidad de ancho de banda, capacidad de almacenamiento entre otros (Garcia Sanjuan, 2009).

2.1.1.2 ARQUITECTURA

Dentro de las arquitecturas presentes en las redes de extremo a extremo se puede encontrar la ***Centralizada*** en esta existe un “servidor” central que mantiene una base de datos con información de los archivos disponibles servidos por cada equipo de la red, cada vez que había un cambio, por ejemplo, cada vez que uno de los equipos conectaba o se desconectaba la base de datos del “servidor” central se actualizaba.

El “servidor” central una vez recibe alguna solicitud de algún archivo este revisa su base de datos y envía las solicitudes equipo destino. Una vez se les informa a ambos equipos (origen-destino) se contactan y se envía el archivo solicitado (Garcia Sanjuan, 2009).

Esta arquitectura es de muy alto rendimiento si se desea localizar recursos, aunque esto depende directamente de que el “servidor” central tenga una muy buena capacidad y velocidad de procesamiento. Como ejemplo de esta arquitectura tenemos a Napster.

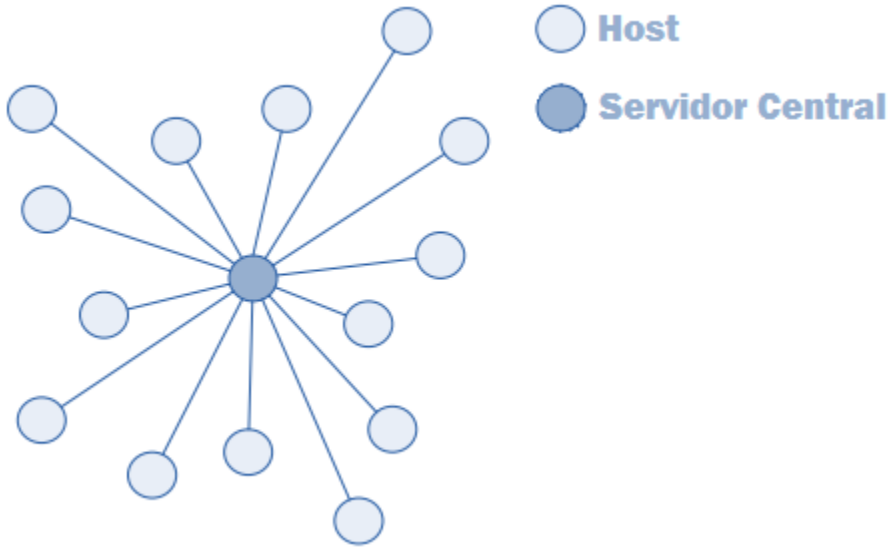


Figura 1 Arquitectura Centralizada Redes Extremo a Extremo fuente (Garcia Sanjuan, 2009).

Otra de las arquitecturas es la ***Descentralizada***, en ella no se tiene ningún “servidor” central y por lo tanto todos los nodos tienen el mismo nivel de importancia, es lo más puro de las redes de extremo a extremo ya que cada equipo puede actuar como servidor y como cliente en la misma red. Todos los equipos conectados entre sí en la red envían constantemente mensaje de peticiones y respuestas y mensajes de control relativamente sencillos como ping entre ellos.

Una de las principales características de esta arquitectura es el uso del tiempo de vida máximo (TTL) para evitar trasmisiones infinitas las cuales por la construcción de esta arquitectura son factibles.

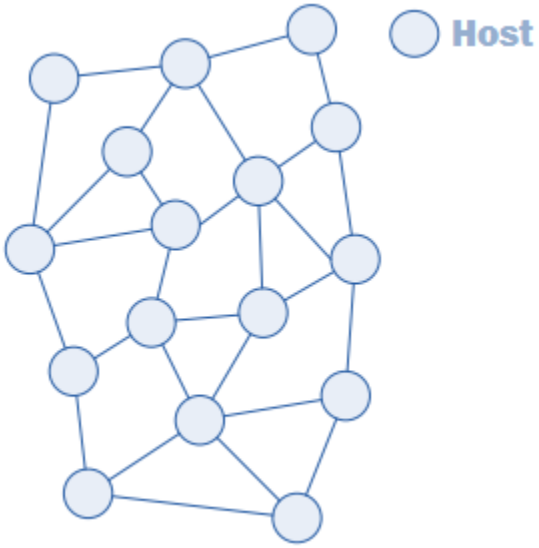


Figura 2 Arquitectura Descentralizada Redes Extremo a Extremo fuente (Garcia Sanjuan, 2009).

Finalmente se puede encontrar una última arquitectura llamada mixta la cual como se puede deducir es una mezcla de las centralizada y la descentralizada en la cual algunos equipos de la red son seleccionados como “servidores” centrales que ayudan a gestionar el tráfico hacia otros equipos.

Una característica de estos “servidores” centrales es que son dinámicos, es decir estos cambian constantemente a medida que más equipos se conectan o desconectan a la red. (Disanzo & Simón, 2006).

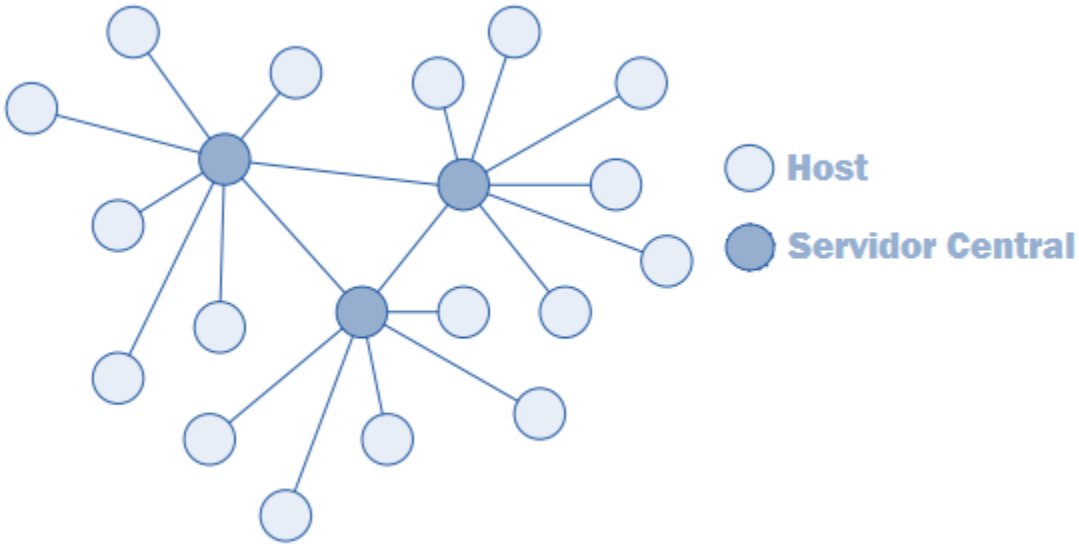


Figura 3 Arquitectura Mixta Redes Extremo a Extremo fuente (Garcia Sanjuan, 2009).

2.1.1.3 APLICACIONES

Dentro de las aplicaciones que las redes de extremo a extremo soportan se encuentran:

- Sistemas para la transmisión y difusión de contenidos multimedia a través de Internet. En estos sistemas, los equipos interesados en el contenido multimedia se conectan a los otros equipos que actuarán como trasmisores para recibir los streams de video y/o audio, esto se diferencia de las trasmisiones convencionales en que los equipos cliente no se conectan con un servidor central de trasmisión basado en IP (Internet Protocol), o IPTV10 (Internet Protocol Television) esto se puede ver en la trasmisiones de emisoras de radio por internet más que todo las de aficionados, trasmisión de videos en vivo o trasmisión de contenido multimedia por ejemplo con ayuda del VLC Player la trasmisión de una película o documental.

- Sistemas de datos distribuidos en los que la información que se desea compartir se almacena en algunos equipos de la red y se configuran para permitir el acceso de otros equipos a esta información, es similar al comportamiento de los servidores

FTP convencionales, se diferencian principalmente en que los equipos que almacenan la información no son servidores FTP como tal si no equipos con carpetas compartidas (Disanzo & Simón, 2006).

- Sistemas de mensajería instantánea y comunicación directa entre equipos. Estos sistemas pueden localizar equipos rápidamente mediante búsquedas principalmente con arquitecturas mixtas, permitiendo una rápida comunicación directa entre equipos y lo que ellos pueda implicar, independientemente del ISP.

- Sistemas de intercambio de archivos. Esta es la aplicación más usada y de hecho la que dio origen a las redes de extremo a extremo. El propósito de esta aplicación es como su nombre lo indica intercambiar archivos entre equipos, se diferencia de los datos distribuidos en que en esta aplicación no hay equipos específicos que almacenan la información si no que cualquier equipo de la red puede contener información que algún otro desea y puede proceder a compartirla, esta información solo sería intercambiada entre los dos equipos interesados a diferencia de la distribuida en donde varios de ellos pueden acceder simultáneamente a la misma información de los equipos que previamente se asignaron como almacenadores de la información.

- Sistemas para el trabajo, edición de archivos en tiempo real de manera cooperativa (Santín González, 2007).

- Sistemas de telefonía por Internet basados en VoIP (Voice over Internet Protocol) los cuales permiten la transmisión en tiempo real de señales de voz por la Red, se diferencian de la VoIP tradicional en que en este caso no hay un servidor principal de VoIP si no que cada equipo del extremo de la conexión realiza las trasmisiones y recepciones respectivas sin tener en cuenta a los demás, el ejemplo más significativo lo tenemos con Skype.

- Sistemas para compartir recursos de CPU y almacenamiento, esto es como “fusionar” varios servidores en uno solo más grande y poderoso que todos ellos vistos de manera individual, es decir si se tiene una cantidad de información que supera la capacidad de almacenamiento de un solo servidor se toma una red mixta, y se conectan varios de ellos lo cuales almacenarán parte de esta información, de igual manera si el procesamiento para una tarea exige una cantidad de hardware que no se dispone se le puede indicar a cada equipo de la red que procese una pequeña parte de la información y una vez lo hayan realizado compartan todos sus resultados a un “servidor” central.

- Sistemas para la ejecución de juegos en forma cooperativa en donde no se usan servidores especializados para la ejecución de juegos como los de Xbox Live o Play Station si no que uno de los jugadores se convierte en el “servidor” principal del juego y los demás jugadores se conectan de manera privada a este como ya se mencionó sin depender de ningún otro tipo de servidor especializado (Santín González, 2007).

2.1.2 TCP

De sus siglas en inglés: Protocolo de Control de Trasmisión es un protocolo que opera en la capa de transporte del modelo TCP/IP, una de sus principales características es que es un protocolo orientado a conexión, esto significa que no inicia una transmisión entre las dos partes hasta que como primera medida esté asegurada la conexión entre estas.

Entre las demás características principales del protocolo TCP vale la pena mencionar que este permite ubicar los datagramas enviados por otros protocolos como el IP en el orden correcto para su correcta interpretación, permite tomar la totalidad de información y dividirla por segmentos de tamaños variables, permite enviar varios paquetes de diferentes fuentes por el mismo canal sin riesgo de que la información se cruce, siempre logra diferenciarlos y finalmente en condiciones normales logra establecer y terminar una conexión entre emisor y trasmisor correctamente (Informatica, 2014).

Este protocolo presenta un sistema con el cual se puede asegurar que los datagramas han sido recibidos, este mecanismo se llama el ACK o Acuse de Recibo en español, este

depende del tiempo que le toma llegar y la cantidad de estos que llegan en un tiempo determinado. Como ya se mencionó es un protocolo orientado a conexión, el equipo emisor toma el rol de cliente, y el equipo receptor toma el rol de servidor, esta comunicación una vez es establecida es full dúplex.

Este protocolo tiene la capacidad de controlar la velocidad de envío de los datos variando el tamaño de los mensajes. Así mismo puede transmitir datos desde varias aplicaciones que se están ejecutando en el mismo equipo: por ejemplo un equipo que este revisando su correo electrónico, mirando videos en streaming, descargando información, y subiendo archivos a una carpeta compartida, todos estos servicios los realiza simultáneamente sin mezclar información o entregarla erróneamente esto lo logra con ayuda de los puertos, para este caso un número de puerto está ligado al tipo de aplicación que se está ejecutando el cual en conjunto con la dirección IP del equipo aseguran tanto el equipo destino del paquete como la aplicación que recibirá esta información (Stallings, 2004).

La importancia del protocolo TCP más exactamente para el presente trabajo es debido a que el protocolo IP no incluye ningún mecanismo de monitoreo de la entrega de los datagramas TCP si garantiza la entrega de la totalidad de estos además de un control de congestión integrado, el cual se explicará más a fondo posteriormente.

Anteriormente se mencionó que el protocolo es orientado a conexión y esto se logra de la siguiente manera: En ambos equipos los puertos TCP deben estar abiertos, la conexión del servidor es pasiva, esto significa que el servidor escucha constantemente hasta recibir una solicitud de conexión.

El cliente solicita un pedido de conexión al servidor en el lugar donde la conexión está abierta, la conexión del cliente en este punto debe ser abierta y activa, el cliente y el servidor deben sincronizarse mediante la denominada negociación en tres etapas (estas también se ven durante el cierre de la conexión) (Tanenbaum, 2003).

En la etapa uno, el cliente transmite un segmento donde el indicador ***SYN*** tiene el valor “1”, esto se hace con el fin de indicar que este es un segmento de sincronización, con un número de secuencia N el cual es llamado número de secuencia inicial del cliente.

En la etapa dos, el servidor recibe el segmento inicial que viene del cliente descrito en la etapa 1 y este le responde con un acuse de recibo, el cual es un segmento en el que el indicador de ACK tiene el valor de “1” y el indicador SYN también tiene el valor de “1”. También se incluye el número de secuencia del servidor, el cual será el número de secuencia inicial para el cliente.

En la etapa tres el cliente transmite un acuse de recibo, el cual es un segmento en el que el indicador ACK tiene el valor de “1” y el indicador SYN tiene el valor de “0”, esto se debe a que ya no es un segmento de sincronización. El número de secuencia presente está incrementado y el acuse de recibo representa el número de secuencia inicial del servidor incrementado en 1, una vez se ha realizado esto los dos equipos (cliente y servidor) están sincronizados y la comunicación puede comenzar (Kurose & Ross , 2004).

2.1.3 VENTANA DESLIZANTE

La ventana deslizante es un método de control del flujo de los datos que se envían entre las partes (emisor, receptor) este control le permite saber al receptor con cuanta velocidad el receptor está procesando los paquetes enviados.

Cada vez que el trasmisor envía un paquete debe esperar hasta recibir una respuesta por parte del receptor esto podía tomar un tiempo relativamente grande por lo que surge la idea de ventana.

Esta ventana consiste en enviar no de manera individual paquetes de trasmisión si no que envía un grupo de estos una vez recibe la confirmación de recepción por parte del receptor envía nuevamente otro grupo de paquetes.

Según como el receptor confirme la llegada de paquetes el trasmisor sabe si debe reducir el tamaño de la ventana y por lo tanto el número de paquetes que envía simultáneamente o si por el contrario puede ampliar la ventana de allí el que sea “deslizante” (Feit, 2004).

2.1.4 ACK (ACKNOWLEDGEMENT)

Como se conoce en español “acuse de recibo” es un método por el cual el receptor le informa al trasmisor que el paquete enviado fue recibido exitosamente.

Cuando un trasmisor envía uno o varios paquetes por cada uno espera un ACK, internamente cada vez que envía un paquete inicia un cronómetro el cual se detiene en el momento en que ha recibido el ACK, aun así ese cronómetro no tiene un tiempo de espera ilimitado.

Si el ACK es recibido dentro de un rango de tiempo que lo clasifica como “rápido” el trasmisor puede aumentar el tamaño de la ventana, si el ACK es recibido dentro de un rango de tiempo que lo clasifica como “lento” el trasmisor procede a reducir el tamaño de la ventana, finalmente si el ACK no llega dentro de los rangos de tiempos preestablecidos se asume que el paquete se perdió y procede a reenviarlo iniciando un nuevo cronómetro y queda a la espera de su respectivo ACK (Corner, 2000).

2.1.5 CONGESTIÓN DE RED

En el mundo de las redes de internet se habla de congestión cuando existe una cantidad de paquetes en tránsito que supera la máxima cantidad de paquetes que pueden ser almacenados en los buffers del receptor o de algún nodo intermediario que re direcciona dichos paquetes.

También está ligado a que la cantidad de usuarios concurrentes sea mayor que la que la capacidad de procesamiento de la red pueda soportar.

Y finalmente se puede dar el caso de que las dos principales causas se congestión se presenten de manera simultánea, que la cantidad total de usuarios concurrentes sea muy alta y que a si vez cada uno de ellos genere una gran cantidad de información.

Eso es clara mente un problema que afecta la calidad del servicio de internet con aspectos como lentitud, aumento de tiempos de envío y recepción, el reenvío de paquetes, entre otros (Tanenbaum, 2003).

2.1.6 ALGORITMO DE CONTROL DE CONGESTIÓN

Un algoritmo de control de congestión se define como el conjunto de técnicas, metodologías y procesos que buscan detectar y solucionar los problemas que surgen cuando el tráfico que circula en una red supera el total del permitido que puede ser procesado correctamente bajo factores como tiempos de arribo, tiempo de procesamiento, tamaño del buffer entre otros y que claramente afectarían la calidad del servicio (Kuchler & Schapranow, 2005).

2.2 MARCO TEÓRICO

2.2.1 CONGESTIÓN EN DIFERENTES ETAPAS DEL INTERNET

La internet o red de redes es un conjunto de redes que están interconectadas entre sí con el fin de compartir información, inicialmente surge como un proyecto de defensa de los Estados Unidos como plan de contingencia durante la guerra fría para lograr mantener interconectados los principales puntos del país en caso de ataques por parte de la entonces URSS.

Continuando con los orígenes de la internet se establece como antecesor el enlace establecido por varias universidades de Estados Unidos durante la década de los 70´s conocido como ARPANET (Advanced Research Projects Agency Network) (Keefer & Baiget, How it all began: a brief history of the Internet, 2009).

Dos décadas después hacia los 90´s dejaría de ser de uso exclusivo de fines militares y educativos y se convertiría en un espacio público para que la gente del común tuviera acceso a todo tipo de información, este fenómeno generó la World Wide Web y su muy conocido "WWW" que más que una unidad es un conjunto de protocolos que permite la consulta de información.

Desde sus inicios ARPANET permitió el envío de correo electrónico y es así como en 1971 se realiza el primer envío de uno de estos, lo cual claramente indicó el inicio de una nueva era lo que se observaría más claramente décadas después cuando el hecho de tener una cuenta de correo electrónico en algo socialmente "obligatorio".

Y es desde este punto desde los 80`s que internet empezó a tener los primeros problemas serios de congestión de tal manera que en 1988 surge un algoritmo de control de congestión llamado TCP Tahoe que principalmente buscaba solucionar los primeros problemas que se encontraron cuando la implementación de internet inicio formalmente, haciendo que la red de ese momento (ARPANET) empezara a utilizar en sus transmisiones el protocolo TCP/IP, detectando el estado de la red y realizando un muy buen control de flujo lo cual logró reducir el número de paquetes perdidos e implementando nuevos términos como inicio lento (Slow-Start) y Retrasmisión Rápida (Fast Retransmit) (Practicas de Redes, Protocolo TCP, 2003). Este funcionó por un tiempo, pero se encontró que su principal inconveniente consistía en que cuando el algoritmo detectaba que los paquetes se perdían (por vencimiento del temporizador) reducía el tamaño de la ventana a su valor mínimo lo cual claramente significaba una reducción en los tiempos de recuperación (Salazar, 2007).

En 1990 surge el algoritmo de control de congestión llamado TCP RENO el cual en principio cumple con las características de su antecesor el TCP Tahoe incluyendo un algoritmo de recuperación rápida el cual trabaja paralelamente con el algoritmo de retrasmisión rápida (Jitendra Padhye, 2000). Una ventaja de este algoritmo es que entraba en estado de retrasmisión rápida una vez recibía múltiples paquetes duplicados aun así el principal inconveniente de TCP RENO consiste en que cuando se ve obligado a reducir la ventana de congestión lo hace múltiples veces (Bogdan Moraru, 2001).

Ya propiamente en internet aproximadamente en el año 1994 aparecería el comercio electrónico algo totalmente nuevo lo cual trajo miles de nuevos usuarios interesados en la facilidad de realizar compras o pagos por internet (Clark & Kleinrock , 2009).

En 1996 aparece el algoritmo TCP NEW RENO, el cual como se puede deducir de su nombre es una modificación del algoritmo TCP RENO, su principal mejora con respecto a su antecesor consiste en su capacidad de detectar múltiples pérdidas de paquetes de manera simultánea. Este algoritmo al igual que su antecesor inicia un estado de retransmisión rápida una vez ha recibido múltiples ACK duplicados, la diferencia se encuentra en que NEW RENO no sale de ese estado de retransmisión rápida hasta que recibe ACK para todos los datos que estaban pendientes en el momento en que entró a el estado ya mencionado, de esta manera es como supera el principal problema de su antecesor el cual reduce la ventana de congestión múltiples veces. Y aunque este algoritmo presenta grandes mejorías no es perfecto, TCP NEW RENO presenta el hecho de que le toma un Round-Trip Delay Time independiente para detectar cada pérdida de paquetes, además de que solo puede deducir una pérdida de paquetes una vez ha recibido un ACK por el primer segmento retransmitido (Practicas de Redes, Protocolo TCP, 2003).

Este algoritmo surge entre 1994 y 1995, su principal característica radica en lo relevante que es el retardo de los paquetes sobre los métodos anteriores los cuales buscan sacar provecho de la pérdida de paquetes con el fin de determinar a qué tasa se deben enviar los paquetes a transmitir (Brakmo, O'Malley, & Peterson, 1994). Este protocolo funciona aumentando el tamaño de la ventana hasta que detecta pérdidas de paquetes debido a problemas de congestión, una vez esto ocurre el algoritmo revisa el Round-Trip Delay Time en todos los segmentos, si este Round-Trip Delay Time es grande al algoritmo asume que la red está congestionada y por lo tanto disminuye el tamaño de la ventana, por consiguiente si detecta que el Round-Trip Delay Time es pequeño se asume que la red no presenta congestión y por lo tanto puede aumentar el tamaño de la ventana (Hengartner, Bolliger, & Gross, TCP Vegas Revisited, 2000).

En 1996 surge otro algoritmo de control de congestión basado en el algoritmo TCP NEW RENO llamado TCP SACK, y este se concentra en trabajar sobre las fallas de sus predecesores el TCP RENO y TCP NEWRENO en lo que se refiere a las múltiples pérdidas de paquetes, así como la retransmisión de más de un paquete perdido por Round-Trip Delay.

Este algoritmo aún conserva el inicio lento e incluye un timeout para los ACK más grande que sus predecesores, una vez el receptor ha detectado que se ha perdido algún paquete debido a que recibe los que le preceden como indica este algoritmo envía un ACK duplicado intencionalmente para avisarle al trasmisor de tal manera que este pueda estar en capacidad de saber cuáles paquetes se han perdido y cuales han llegado correctamente para retrasmitir solo los necesarios (Ekiz, Rahman, & Amer, 2011).

De manera paralela este algoritmo realiza constantemente una estimación del tráfico pendiente que tiene la red si ve que el tráfico pendiente es grande reduce el tamaño de la ventana del trasmisor, si por el contrario detecta que es pequeño aumenta el tamaño de la ventana del emisor, pero siempre manteniendo su ventana más pequeña que la ventana de congestión.

Hacia 1996 aparecen los primeros celulares que contenían incorporados en sus firmwares el protocolo la pila de protocolos TCP/IP y por lo tanto tendrían la capacidad de conectarse a internet lo cual, aunque en este momento no indicó un aumento significativo de usuarios, si sentó las bases de los que más adelante sería una integración tecnológica que si implicó un aumento significativo de usuarios y por lo tanto tráfico.

Una última solución de las múltiples existentes es la del algoritmo de control de congestión llamado TCP CUBIC, este algoritmo como principal característica mantiene el sincronismo de los ACK para poder incrementar el tamaño de su ventana además que cuando el algoritmo entra en la etapa de recuperación rápida no realiza cambios en ninguna de sus etapas.

Este algoritmo empieza su operación al detectar una pérdida de paquetes en ese momento toma el tamaño actual de la ventana como el tamaño máximo y luego inicia un decremento multiplicativo del tamaño de la misma para luego entrar en una etapa de recuperación en donde el algoritmo empieza a aumentar el tamaño de la ventana de manera cúbica hasta alcanzar el valor máximo que tomó al momento de haber detectado la pérdida (Rhee & Xu, 2008).

En 1998 aparece google, el buscador más famoso del mundo que sería la herramienta que le permitiría a millones de usuarios de todo el mundo poder navegar con mayor facilidad en lo que respecta a una pregunta sumamente frecuente: ¿dónde puedo buscar la información que necesito?

En 1999 aparecen los blogs, espacios donde miles de usuarios compartían sus ideas y experiencias lo cual aumentaría el tiempo de uso de las redes por parte de estos usuarios, aspecto fundamental para la investigación del proyecto.

En el año 2003 aparece Skype y su posibilidad de realizar desde la perspectiva del usuario llamadas telefónicas pero por internet así como video llamadas por el mismo medio, algo que generaría un aumento de la intensidad de tráfico ya que estos tipos de servicios necesitan prestarle al usuario una experiencia en tiempo real aún más exigente que otros tipos de datos a trasmitir.

En el año 2004 aparecen las primeras redes sociales, sucesoras de los blogs que actualmente son una fuente sumamente grande de usuarios concurrentes cada uno con su respectivo tráfico algunos inclusive de 24 horas al día (Cohen-Almagor,, 2011).

Hacia el año 2005 aparece la que sería la plataforma de video y streaming más grande del mundo llamada youtube con ella se popularizó a escala global el streaming algo que claramente aumentaría aún más los niveles de tráfico.

Finalmente, hacia el año 2007 es cuando los smartphones integrarían la gran mayoría de los servicios ya mencionados en una única terminal, correo electrónico, navegación en

internet, llamadas usando VoIP, videollamadas, redes sociales, streaming, radio por internet y televisión por internet todo esto al alcance de un dedo, esto provocó un tráfico sumamente alto el cual por su propio tamaño genera constantemente congestión.

2.2.2 INVESTIGACIONES PREVIAS CON ALGORITMO TCP RENO Y TCP VEGAS

En el año 1995 Lawrence S. Brakmo, Larry L. Peterson (Lawrence & Peterson, 1995) realizaron una investigación acerca del estado el arte del algoritmo de control de congestión TCP VEGAS, el cual tiene un mejor rendimiento que el TCP RENO, según datos de los autores este es mejor entre un 37% y un 71% además de solo presentar desde un quinto hasta un medio de pérdidas de paquetes.

Los autores muestran las 3 técnicas principales que el algoritmo TCP VEGAS implementa y presentan los resultados de sus experimentos utilizando una comparativa exhaustiva entre las simulaciones y mediciones reales en internet.

Dentro de sus principales hallazgos encontraron que se debe incluir un nuevo mecanismo de tiempo de espera así como una nueva metodología que evite la congestión en donde se debe tratar de controlar el número de buffers adicionales que ocupan la conexión en la red y finalmente un mecanismo de comienzo lento modificado (Lawrence & Peterson, 1995).

2.2.3 INVESTIGACIONES PREVIAS CON ALGORITMO TCP NEWRENO

En el año 2014 Satoshi Utsumi, Salahuddin Muhammad Salim Zabir (Utsumi, Muhammad , & Zabir, 2014) realizaron una investigación acerca de nuevos algoritmos de control de congestión que se concentran en el retardo principalmente el Caia-Hamilton (CHD) y el Caia Delat-Gradient (CDG) los cuales han tomado un lugar importante en el mundo del control de tráfico debido a su eficiencia en redes cableadas e inalámbricas y a su vez mencionan que el éxito de estos dos algoritmos radica en que fueron diseñados para trabajar a la par con TCP NewReno.

Estos protocolos (Caia-Hamilton (CHD) y Caia Delat-Gradient (CDG)) basan su mecanismo de actualización de paquetes en retardos lo que hace que sean derrotados cuando coexisten con TCP NewReno, este último presenta una cantidad de flujos de datos de gran magnitud de manera simultánea, además que específicamente para el caso del Caia-Hamilton (CHD) es bastante complicado seleccionar los valores adecuados en lo que respecta a sus ajustes para que presenten el rendimiento esperado.

Para solucionar esto los autores de la investigación propusieron un mecanismo nuevo que superara las falencias ya mencionadas y este mecanismo lo nombraron: Wireless Friendly Congestion Control (WFCC) para ello como primera medida buscaron diseñar un esquema de control de congestión que fuera amigable con TCP NewReno en especial cuando ambos fluyen simultáneamente a través de enlaces inalámbricos y que fuera robusto frente a los errores típicos de enlace bajo una amplia gama de espacio de búfer en la red.

Para ello llevaron a cabo múltiples simulaciones y emulaciones los cuales mostraron que Wireless Friendly Congestion Control (WFCC) puede conducir hasta una mejora del 250% de rendimiento en comparación con los esquemas y TCP NewReno además de un rendimiento excelente a la par de TCP NewReno cuando ambos fluyen de manera simultánea a través de enlaces inalámbricos (Utsumi, Muhammad , & Zabir, 2014).

2.2.4 INVESTIGACIONES PREVIAS DE CONGESTIÓN EN REDES EXTREMO A EXTREMO

En el año 2014 Cheng Cui , Lin Xue , Chui-Hui Chiu , Praveenkumar Kondikoppa , Seung-Jong Park (Cui, Xue, Chiu, Kondikoppa, & Park, 2014) realizaron una investigación en uno de los problemas que se presentan muy a menudo en redes tipo LAN-WAN-LAN y es el tamaño del buffer de los diferentes routers por los que puede pasar un paquete, este problema ha sido tema de discusión durante ya largo tiempo en donde se discute como mejorar los enlaces en las redes de alta velocidad en el que el tamaño del buffer de los routers es ideal y por ser routers de alta velocidad se podría reducir el tamaño del buffer.

Los principales problemas a atacar según Cheng Cui , Lin Xue , Chui-Hui Chiu , Praveenkumar Kondikoppa , Seung-Jong Park (Cui, Xue, Chiu, Kondikoppa, & Park, 2014) vendrían siendo 3: el exceso de ancho de banda, el tener ráfagas de datos en lugar de flujos y el que los paquetes sean asíncronos, aun así las investigaciones realizadas sugieren que estos 3 aspectos podrían no ser suficientes y que el tráfico TCP aun seguiría sufriendo retrasos y problemas con routers de buffers pequeños.

Para solucionar esto Cheng Cui , Lin Xue , Chui-Hui Chiu , Praveenkumar Kondikoppa , Seung-Jong Park (Cui, Xue, Chiu, Kondikoppa, & Park, 2014) proponen un algoritmo de control de congestión al que llamaron: “multicanal desincronizado TCP” el cual consiste en crear flujos con varios canales paralelos los cuales tienen total independencia entre sí en cuanto a sincronización adaptándose a las necesidades de los routers evitando problemas de tráfico principalmente en las ráfagas de datos.

En las simulaciones que ellos realizaron se encontró que aunque sigue existiendo cuellos de botella con el algoritmo de “multicanal desincronizado TCP” el cuello de botella en ráfagas de datos es de menor magnitud en un 80% comparado con los demás algoritmos de control de congestión en redes netamente ópticas (Cui, Xue, Chiu, Kondikoppa, & Park, 2014).

2.2.5 INVESTIGACIONES PREVIAS DE MÉTRICAS DE CONTROL DE CONGESTIÓN

En el año 2012 K. Avrachenkova, U. Ayestab, J. Doncelc, P. Jacko (Avrachenkova, Ayestab, Doncelc, & Jacko, 2012) realizaron una investigación enfocada en el problema existente en trasmisiones rápidas de flujos entre diferentes routers que pueden estar presentes en redes así como la congestión que estos sufren, para ello se realizó el modelamiento de la interacción entre fuentes de información TCP y un router aplicándole un cuello de botella con el objetivo de diseñar controles óptimos de procesamiento de paquetes en cola del router.

Los autores en esta investigación decidieron centrarse en una versión sencilla del problema obtenida mediante la disminución de la capacidad del buffer la cual aún se satisfacía en todo momento, ellos lograron observar que esta versión sencilla les permitía reducir el problema de los multi-flujos, y con esto poder analizar matemáticamente la existencia de políticas de control de congestión.

Teniendo en cuenta que una gran variedad de parámetros que tienen los flujos de paquetes TCP son posibles de controlar directamente en los routers configurando el índice de políticas, pero hay que recordar que no siempre las políticas del canal son controlables, las simulaciones ellos las implementaron en el Network Simulator-3 en donde variaron el índice de políticas implementándolo en una topología sencilla con el fin de determinar si era aplicable en redes reales.

Una vez realizaron las simulaciones estas arrojaron que el índice de políticas puede lograr una amplia gama de propiedades ajustables en para todas las versiones de TCP, con parámetros como cantidad de usuarios, diferentes tiempos de ida y vuelta de los paquetes y el buffers mínimos requeridos para lograr la plena utilidad de la cola (Avrachenkova, Ayestab, Doncelc, & Jacko, 2012).

En el año 2012 Ghassan A. Abed, Mahamod Ismail, Kasmiran Jumari (Abed, Ismail, & Jumari, 2012) realizaron una investigación para llevar a cabo un análisis a profundidad de el por qué han surgido diversas técnicas de control de congestión, en una primera parte se justifica el por qué se sigue usando TCP aunque esté presente una clara congestión, ellos afirman que TCP es el estándar más confiable de toda la pila de protocolos de la internet que junto con el protocolo IP se complementan de manera ideal la mayoría de las veces, es por esto que aunque son toda una pila de protocolos normalmente se le conoce como TCP/IP, esta combinación presenta una confiabilidad sumamente alta en los envíos de paquetes y flujos constantes en redes tipo extremo a extremo por lo cual aunque pueda generar congestión se sigue usando.

En la segunda parte los autores evidencian que el protocolo TCP no es usado en la totalidad de aplicaciones WEB, es normalmente usado en servicios como correo electrónico,

transferencia de archivos, administración, gestión y control remoto, aun así existen otras aplicaciones que por su misma naturaleza no requieren un servicio de flujo constante de datos fiables y por lo tanto pueden usar el protocolo UDP, el cual proporciona un servicio de datagramas que es mucho más eficaz en cuanto a velocidad de trasmisión pero con una gran posibilidad de errores y necesidades de retrasmisión.

En la tercera parte los autores mencionan que un método para controlar la congestión podría ser el determinar el ancho de banda de cada segmento de la red y administrarlo, pero hacen hincapié en que es una tarea sumamente tediosa y complicada, es por esto que surgieron los algoritmos de control de congestión.

Finalmente los autores realizan un análisis de como es el control de congestión propio de TCP y de esto saca las conclusiones de cuáles deben ser las principales características que se necesitan para poder diseñar y desarrollar efectivamente un nuevo algoritmo de control de congestión (Abed, Ismail, & Jumari, 2012).

2.3 ESTADO DEL ARTE

2.3.1 PRUEBAS DE ESTRÉS

Para poder verificar el estado del arte de la congestión en redes de extremo a extremo en escenarios LAN-WAN-LAN se realizaron una serie de pruebas de estrés en infraestructura real durante un total de 5 días para validar los niveles de congestión que una red de alrededor de 2400 usuarios.

Dichas pruebas de estrés fueron realizadas sobre los principales exponentes de los servicios actuales de internet:

- **Google.com**: Como servicio de buscador web.
- **Youtube.com:** Como servicio de streaming y/o trasmisión de video.
- **Eltiempo.com**: Como servicio de noticias e información.
- **Caracoltv.com**: Como servicio de streaming, ocio y entretenimiento.

- **Facebook.com**: Como servicio de red social.

2.3.2 METODOLOGÍA

Se tuvo en cuenta uno de los parámetros más importantes de las redes de extremo a extremo, como lo son la cantidad y el tipo de nodos entre los extremos, así como las líneas de transmisión y el ancho de banda de las mismas.

Se aseguró dentro de cada una de las LAN que se encuentran en los extremos de los puntos finales (origen y destino) que en el proceso de comunicación atravesaran la WAN.

Una vez se ha hecho esto se determina el tipo de tráfico, y para cada uno tener en cuenta ancho de banda siendo de 8Mbps, latencia siendo en promedio 115ms y jitter siendo en promedio de 5ms en condiciones normales.

Una vez se realiza esto se determina sobre que topologías se va a simular respondiendo la pregunta, ¿Cuáles son los escenarios más comunes? Para este caso se probó sobre LAN-WAN-LAN, una vez se definen estos se varía el tamaño de los paquetes según comportamientos reales aprendidos de la experiencia estas variaciones se realizaron desde los 12000 bytes (1,333 megabytes) hasta los 310000 bytes (34,444 megabytes).

Después de esto se tienen en cuenta los elementos (swtiches, routers), siendo para cada LAN 3 swtiches convencionales y 1 fronterizo.

En cuanto a seguridad se tienen en cuenta la confidencialidad (toda la información confidencial se envió por MPLS), y cómo se ha de proteger la WAN de ataques externos, para este caso se tiene un Next-Generation Firewall de McAffe.

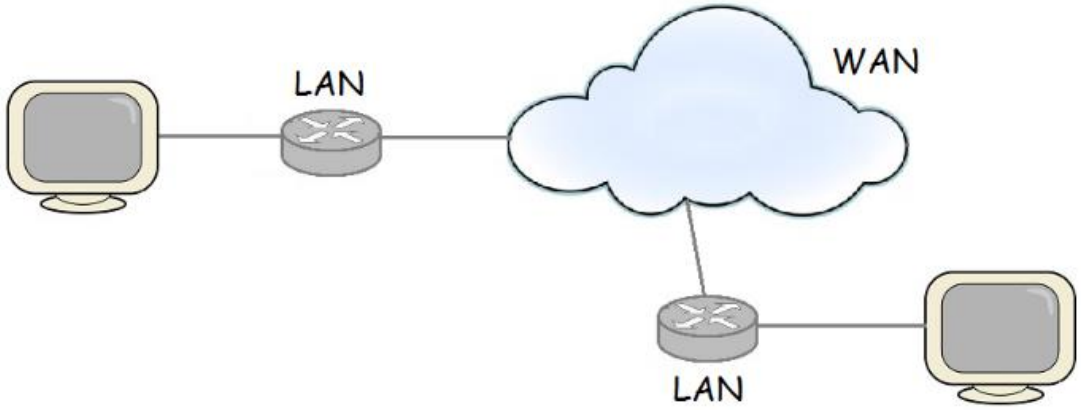


Figura 4 Esquema de red extremo a extremo a probar Fuente: banco de imágenes gratuitas openclipart.org

Entre los nodos se decidió probar diferentes tipos de tráfico variando en todos los casos el tamaño del buffer, de igual manera se variaron los tamaños de segmento y el tiempo de los ACK para así lograr ver su comportamiento según estos cambios.

2.3.3 MODELOS A DISCUTIR

Se realizaron pruebas de estrés con ayuda de la herramienta libre jmeter desde el mismo punto de una red LAN 802.3 hacia 5 servidores pertenecientes a otra red LAN 802.3 pasando en los 5 casos por red WAN siendo los 5 servidores:

- Google.com

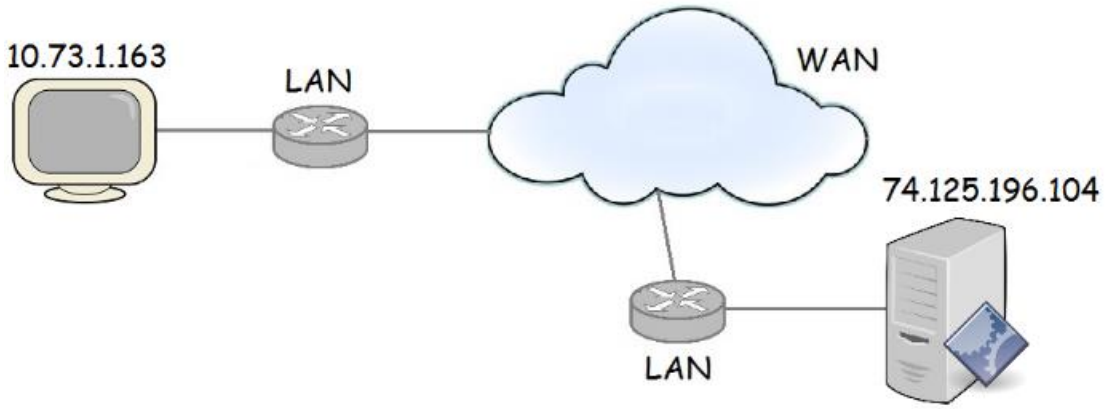


Figura 5 Esquema de red extremo a extremo para la prueba con google.com Fuente: banco de imágenes gratuitas openclipart.org

- Youtube.com

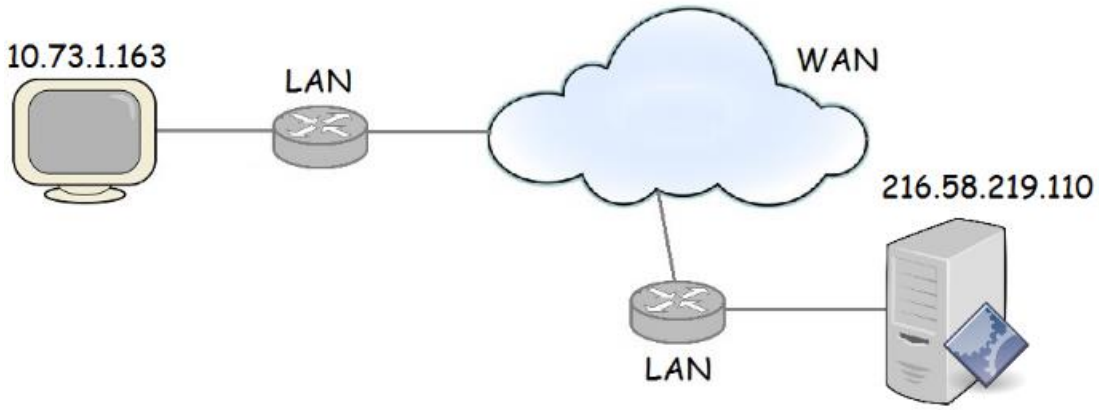


Figura 6 Esquema de red extremo a extremo para la prueba con youtube.com Fuente: banco de imágenes gratuitas openclipart.org

- Eltiempo.com

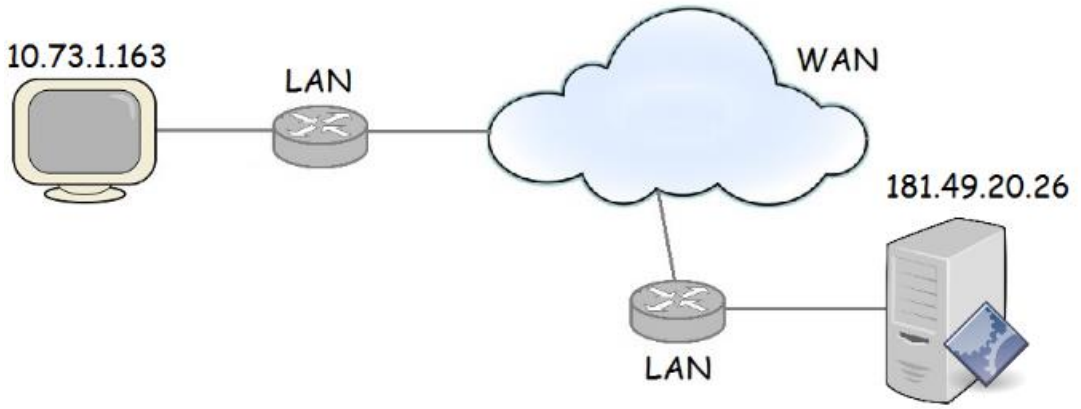


Figura 7 Esquema de red extremo a extremo para la prueba con eltiempo.com Fuente: banco de imágenes gratuitas openclipart.org

- Caracoltv.com

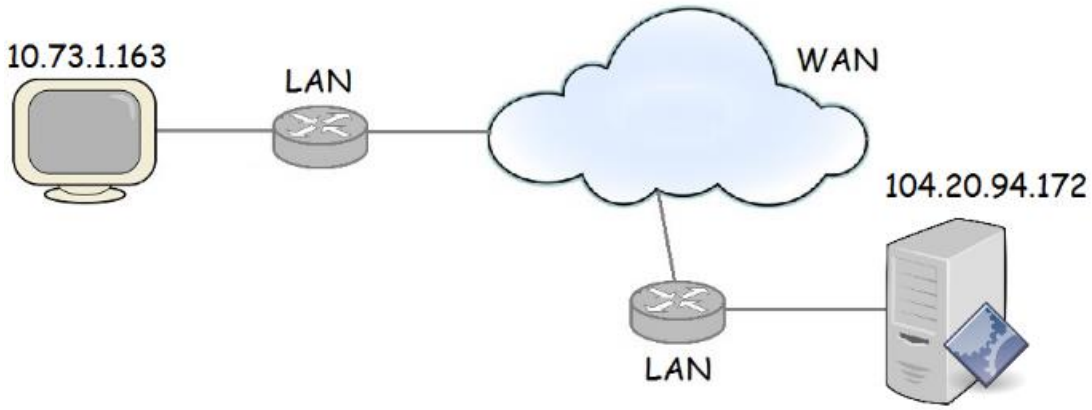


Figura 8 Esquema de red extremo a extremo para la prueba con caracoltv.com Fuente: banco de imágenes gratuitas openclipart.org

- Facebook.com

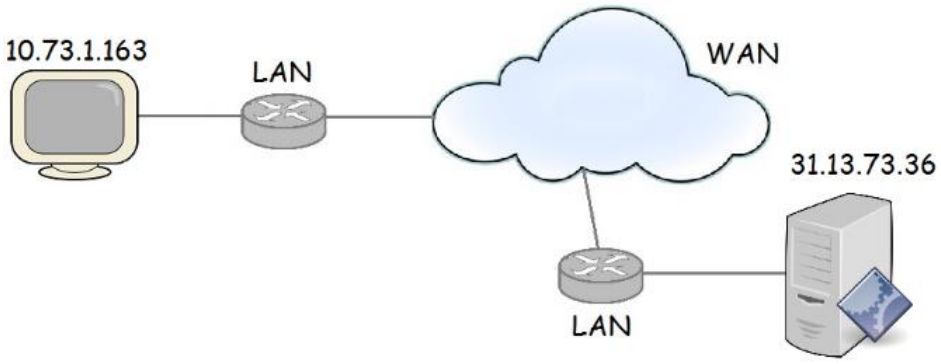


Figura 9 Esquema de red extremo a extremo para la prueba con facebook.com Fuente: banco de imágenes gratuitas openclipart.org

Aumentando el número de usuarios concurrentes con el fin de generar congestión y observar el comportamiento, así como el número total de nodos por el que pasaron los paquetes.

2.3.4 DISCUSIÓN DE RESULTADOS

De las pruebas realizadas durante 5 días a los 5 servidores se encontraron los siguientes resultados:

Google.com

Día	IP Origen	IP Destino	# De Nodos Detectados
1	10.73.1.163	74.125.196.104	16
2	10.73.1.163	74.125.138.106	14
3	10.73.1.163	216.58.192.68	14
4	10.73.1.163	216.58.192.68	14
5	10.73.1.163	74.125.196.147	16

Tabla 1 Datos de los extremos pruebas google.com Fuente: Creación propia resultado de la ejecución de la prueba

Día	Número de Usuarios	Tiempo promedio de los ACK (ms)	Número de conexiones exitosas	Cantidad promedio de bytes enviados	% De error
1	1070	949	1027	12428	4.02
2	1563	3279	1513	12514	3.2
3	1872	941	1822	11926	2.67
4	2175	1150	2126	11971	2.25
5	2581	1071	2534	12017	1.82

Tabla 2 Resultados pruebas de estrés servidor google.com Fuente: Creación propia resultado de la ejecución de la prueba

Youtube.com

Día	IP Origen	IP Destino	# De Nodos Detectados
1	10.73.1.163	216.58.219.110	14
2	10.73.1.163	216.58.192.78	14
3	10.73.1.163	216.58.192.78	14
4	10.73.1.163	216.58.192.78	14
5	10.73.1.163	216.58.192.78	14

Tabla 3 Datos de los extremos pruebas youtube.com Fuente: Creación propia resultado de la ejecución de la prueba

Día	Número de Usuarios	Tiempo promedio de los ACK (ms)	Número de conexiones exitosas	Cantidad promedio de bytes enviados	% De error
1	1076	66874	279	104403	74.07
2	1552	45626	831	89426	46.46
3	1854	49918	968	88348	47.79
4	2152	48237	1036	82424	51.86
5	2552	38390	1692	115479	33.7

Tabla 4 Resultados pruebas de estrés servidor youtube.com Fuente: Creación propia resultado de la ejecución de la prueba

eltiempo.com

Día	IP Origen	IP Destino	# De Nodos Detectados
1	10.73.1.163	184.51.150.139	10
2	10.73.1.163	181.49.20.59	10
3	10.73.1.163	184.51.150.136	10
4	10.73.1.163	184.51.150.91	15
5	10.73.1.163	181.49.20.26	10

Tabla 5 Datos de los extremos pruebas eltiempo.com Fuente: Creación propia resultado de la ejecución de la prueba

Día	Número de Usuarios	Tiempo promedio de los ACK (ms)	Número de conexiones exitosas	Cantidad promedio de bytes enviados	% De error
1	1052	21866	953	309082	9.41
2	1556	13452	1489	169663	4.31
3	1853	10357	1778	171605	4.05
4	2155	18106	2053	171538	4.73
5	2554	7135	2479	140951	2.94

Tabla 6 Resultados pruebas de estrés servidor eltiempo.com Fuente: Creación propia resultado de la ejecución de la prueba

caracoltv.com

Día	IP Origen	IP Destino	# De Nodos Detectados
1	10.73.1.163	104.20.94.172	12
2	10.73.1.163	104.20.94.172	12
3	10.73.1.163	104.20.94.172	12
4	10.73.1.163	104.20.94.172	12
5	10.73.1.163	104.20.94.172	12

Tabla 7 Datos de los extremos pruebas caracoltv.com Fuente: Creación propia resultado de la ejecución de la prueba

Día	Número de Usuarios	Tiempo promedio de los ACK (ms)	Número de conexiones exitosas	Cantidad promedio de bytes enviados	% De error
1	1057	8549	952	118168	9.93
2	1556	8725	1487	122427	4.43
3	1855	7610	1805	74590	2.7
4	2161	8948	2110	71118	2.36
5	2559	3637	2509	58939	1.95

Tabla 8 Resultados pruebas de estrés servidor caracoltv.com Fuente: Creación propia resultado de la ejecución de la prueba

facebook.com

Día	IP Origen	IP Destino	# De Nodos Detectados
1	10.73.1.163	31.13.73.36	15
2	10.73.1.163	31.13.73.36	15
3	10.73.1.163	31.13.73.36	16
4	10.73.1.163	31.13.73.36	15
5	10.73.1.163	31.13.73.36	15

Tabla 9 Datos de los extremos pruebas facebook.com Fuente: Creación propia resultado de la ejecución de la prueba

Día	Número de Usuarios	Tiempo promedio de los ACK (ms)	Número de conexiones exitosas	Tamaño Promedio del segmento (bytes)	% De error
1	1064	5170	1014	69780	4.7
2	1559	4012	1502	70527	3.66
3	1865	4888	1804	70801	3.27
4	2151	4717	2091	71821	2.79
5	2559	4153	2493	71966	2.58

Tabla 10 Resultados pruebas de estrés servidor facebook.com Fuente: Creación propia resultado de la ejecución de la prueba

De los cuales se obtuvieron las siguientes gráficas:

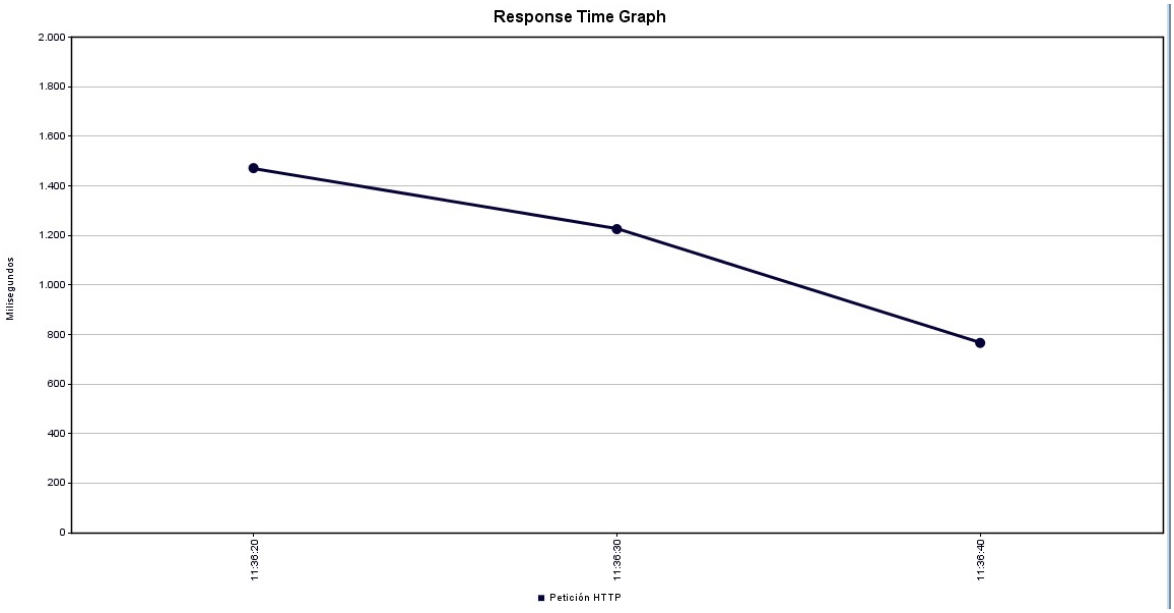


Figura 10 Gráfica de resultados pruebas de estrés servidor google.com día 1 Fuente: Creación del software libre Jmeter como resultado de las pruebas

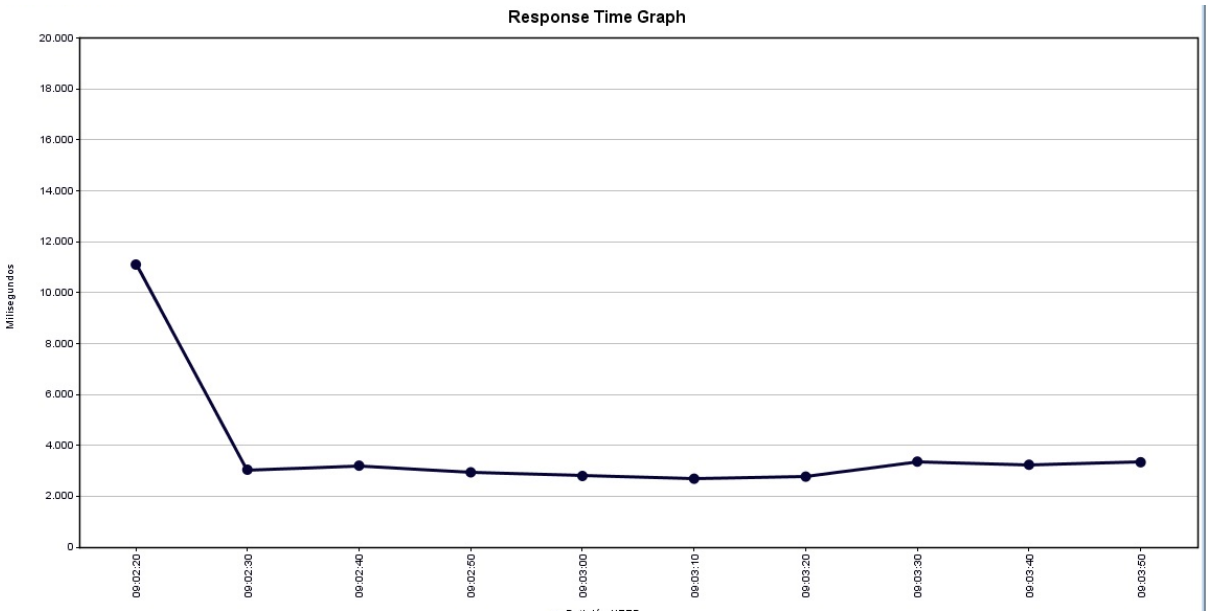


Figura 11 Gráfica de resultados pruebas de estrés servidor google.com día 2 Fuente: Creación del software libre Jmeter como resultado de las pruebas

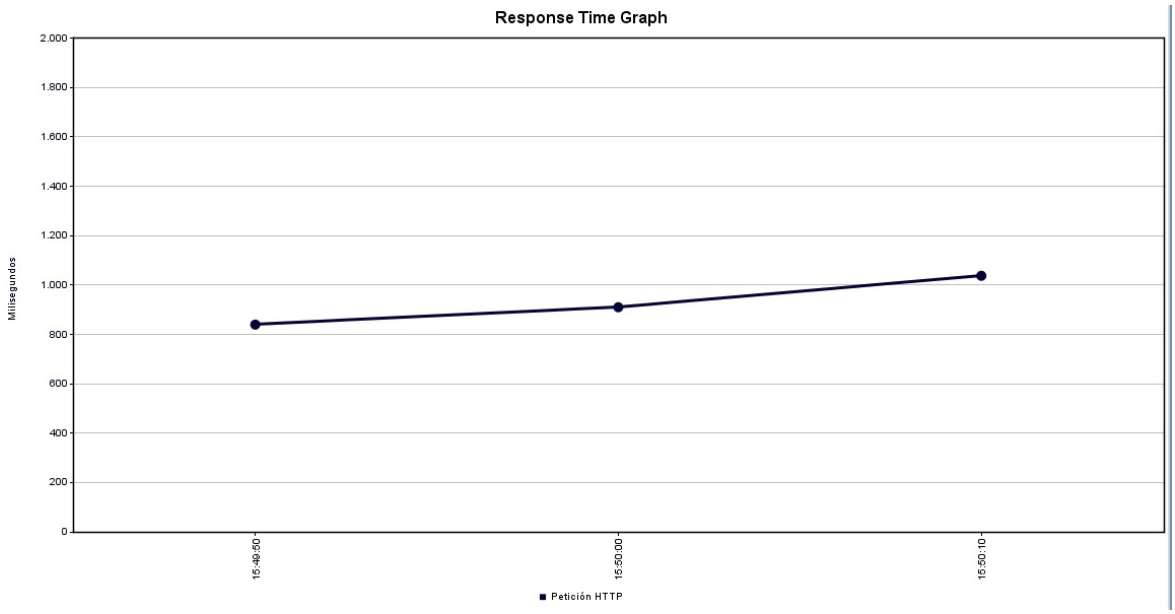


Figura 12 Gráfica de resultados pruebas de estrés servidor google.com día 3 Fuente: Creación del software libre Jmeter como resultado de las pruebas

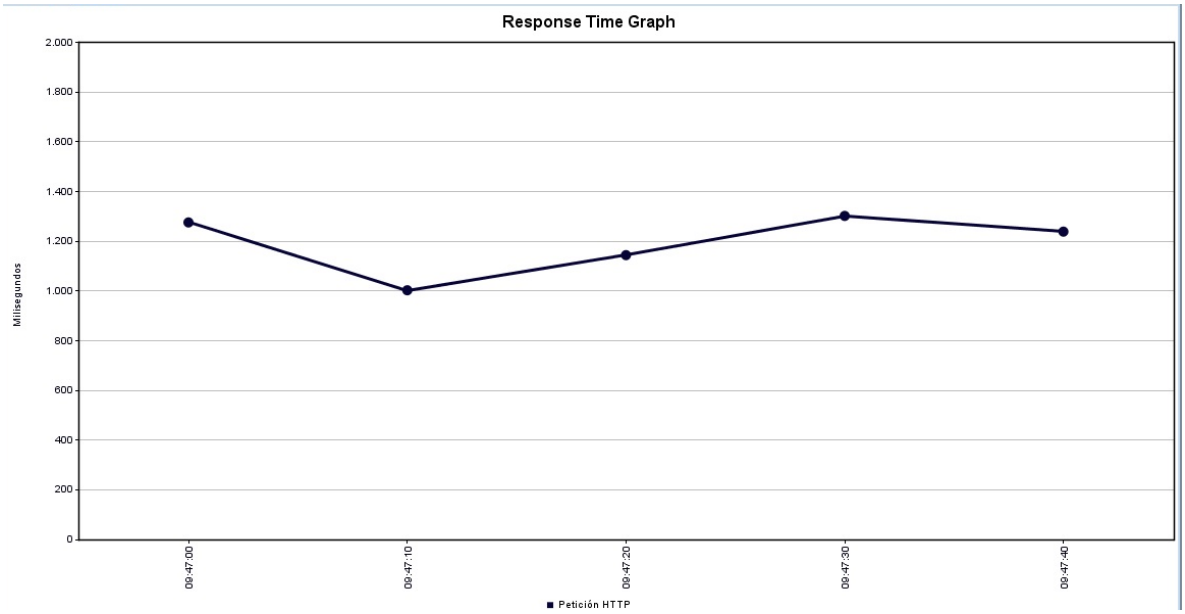


Figura 13 Gráfica de resultados pruebas de estrés servidor google.com día 4 Fuente: Creación del software libre Jmeter como resultado de las pruebas

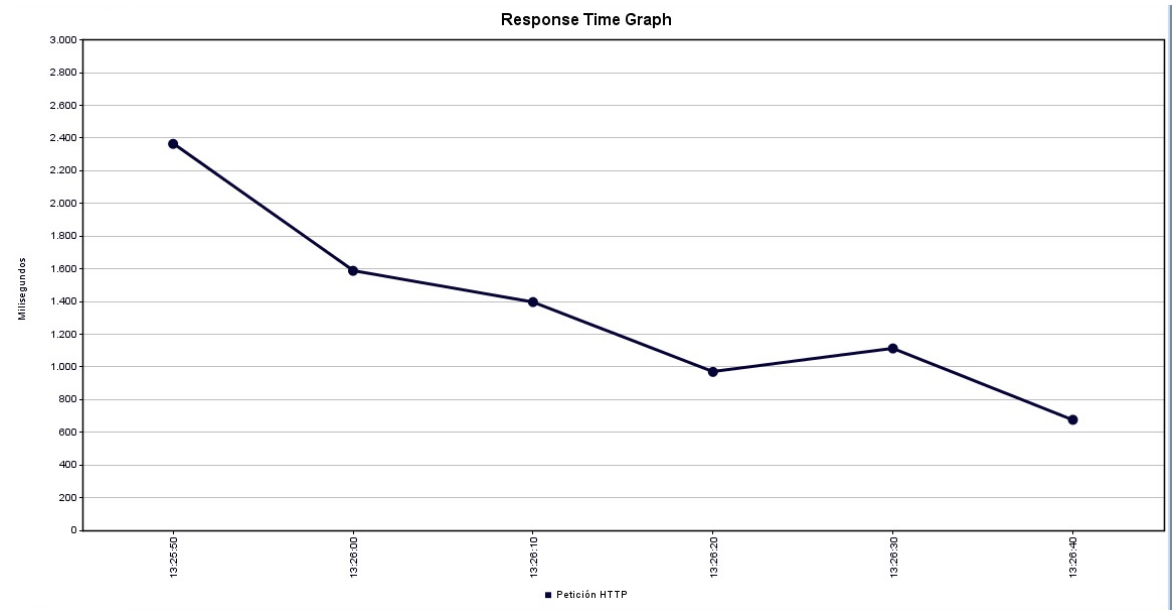


Figura 14 Gráfica de resultados pruebas de estrés servidor google.com día 5 Fuente: Creación del software libre Jmeter como resultado de las pruebas

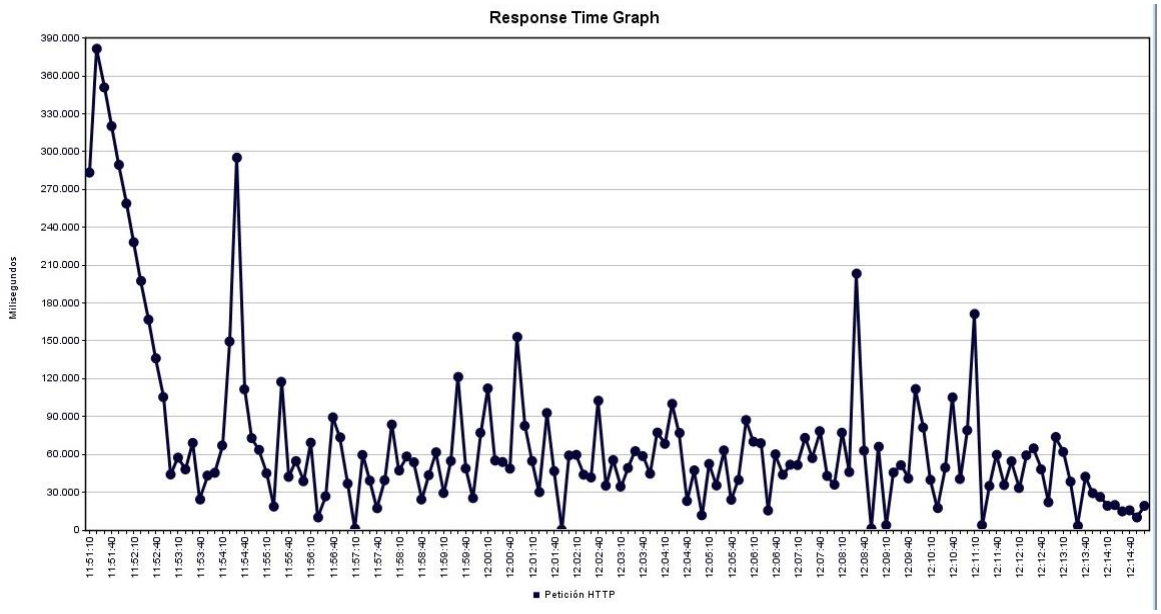


Figura 15 Gráfica de resultados pruebas de estrés servidor youtube.com día 1 Fuente: Creación del software libre Jmeter como resultado de las pruebas

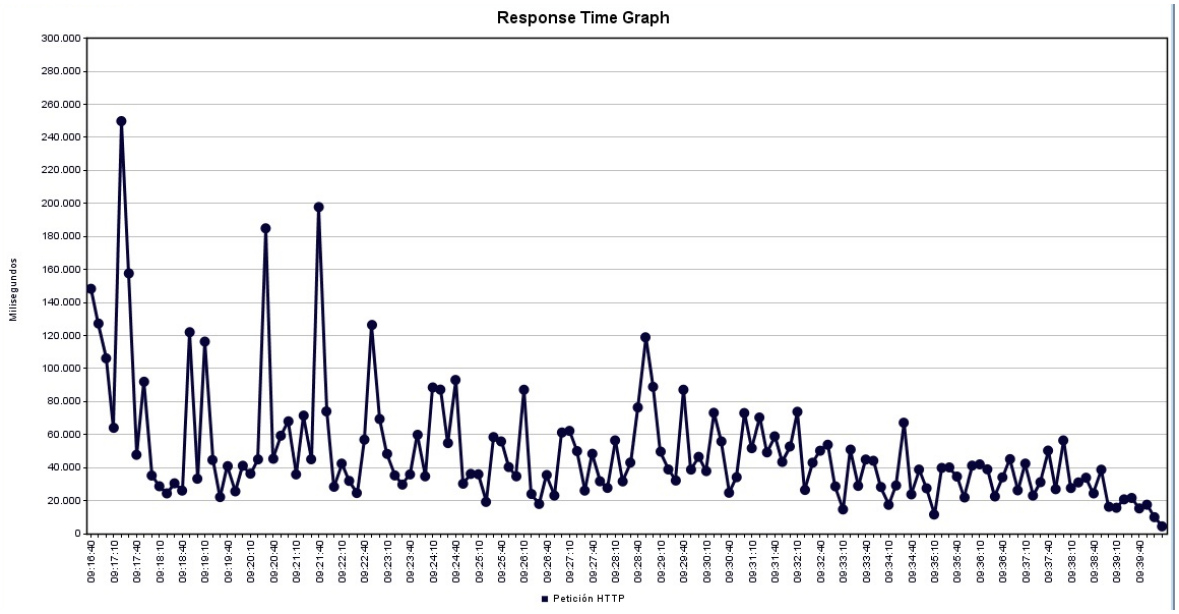


Figura 16 Gráfica de resultados pruebas de estrés servidor youtube.com día 2 Fuente: Creación del software libre Jmeter como resultado de las pruebas

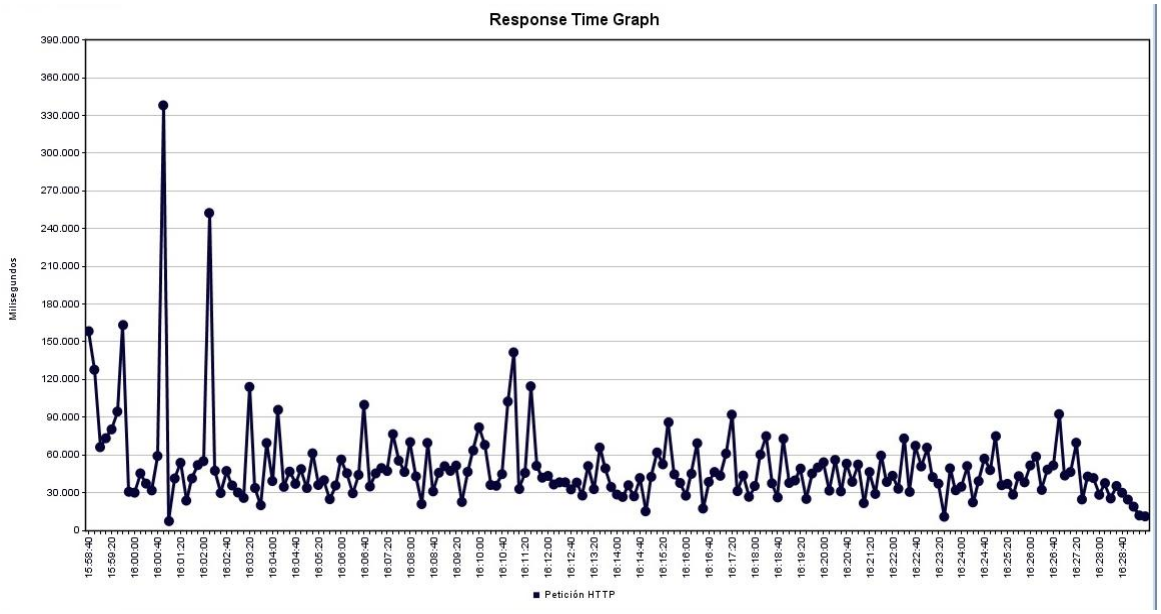


Figura 17 Gráfica de resultados pruebas de estrés servidor youtube.com día 3 Fuente: Creación del software libre Jmeter como resultado de las pruebas

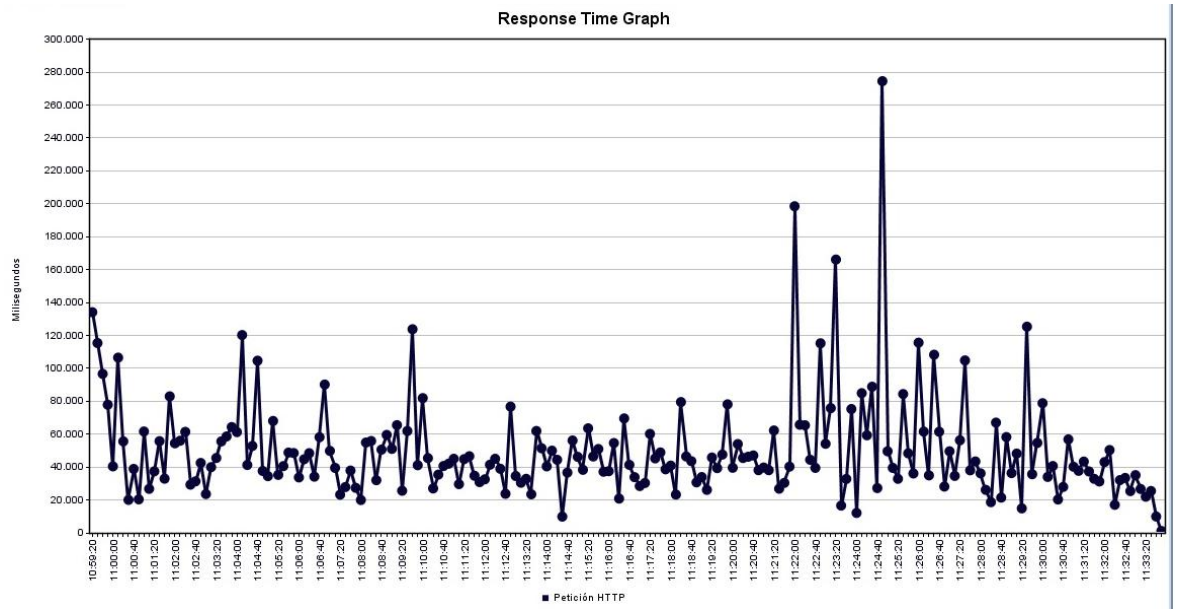


Figura 18 Gráfica de resultados pruebas de estrés servidor youtube.com día 4 Fuente: Creación del software libre Jmeter como resultado de las pruebas

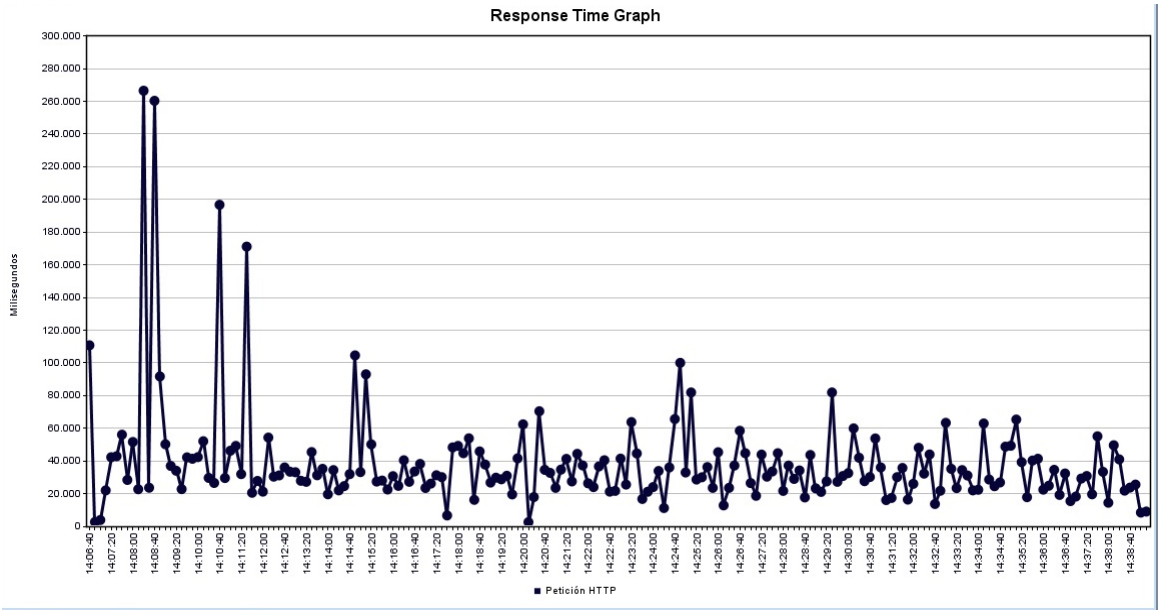


Figura 19 Gráfica de resultados pruebas de estrés servidor youtube.com día 5 Fuente: Creación del software libre Jmeter como resultado de las pruebas

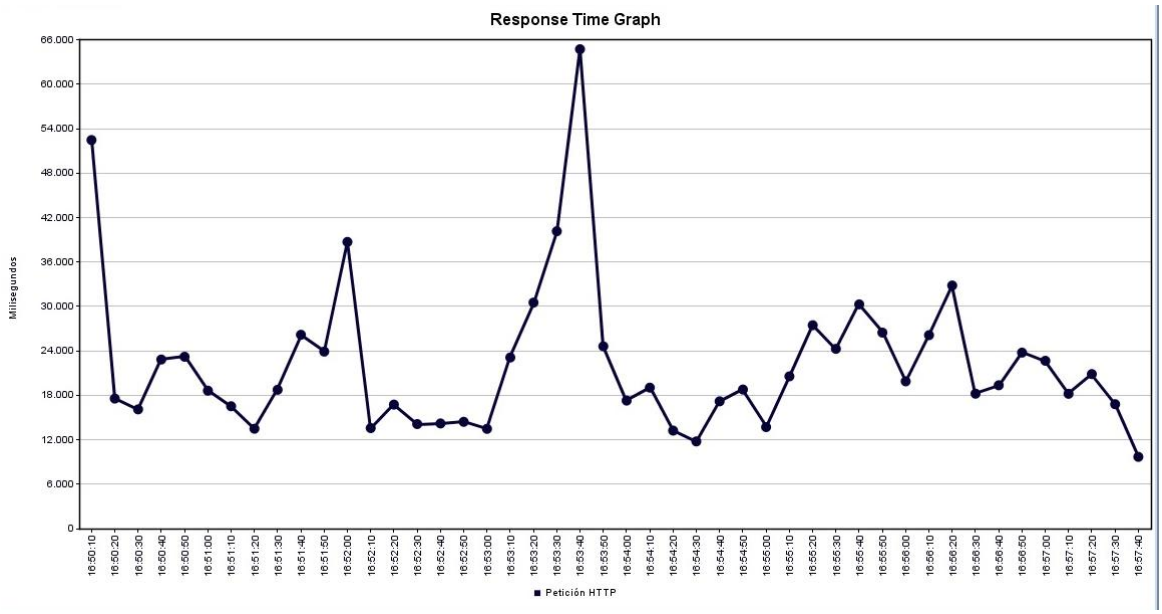


Figura 20 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 1 Fuente: Creación del software libre Jmeter como resultado de las pruebas

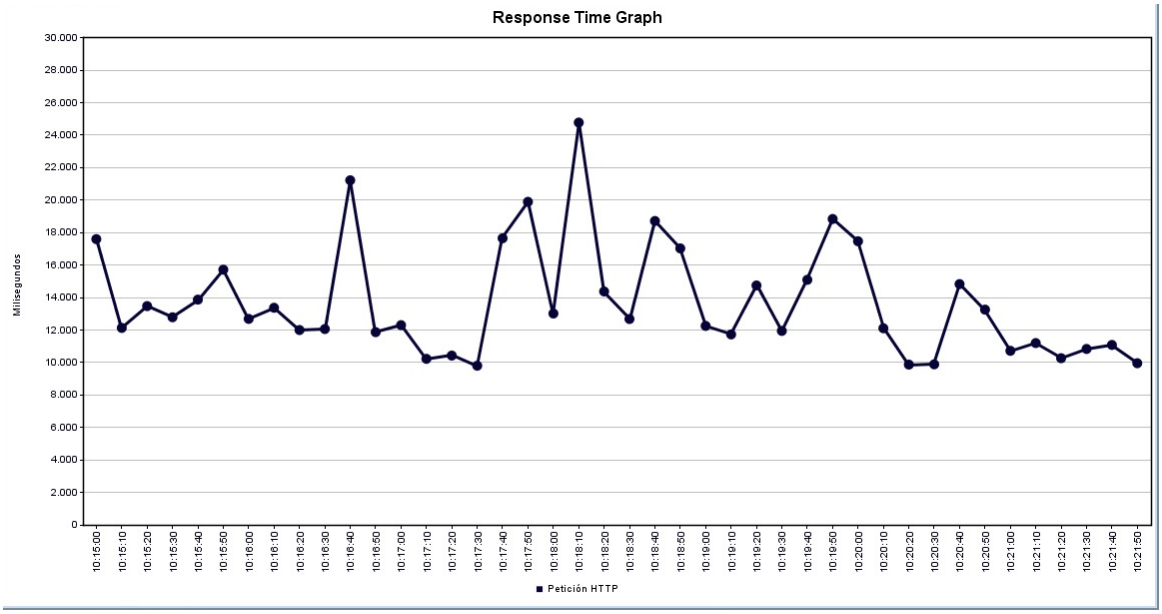


Figura 21 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 2 Fuente: Creación del software libre Jmeter como resultado de las pruebas

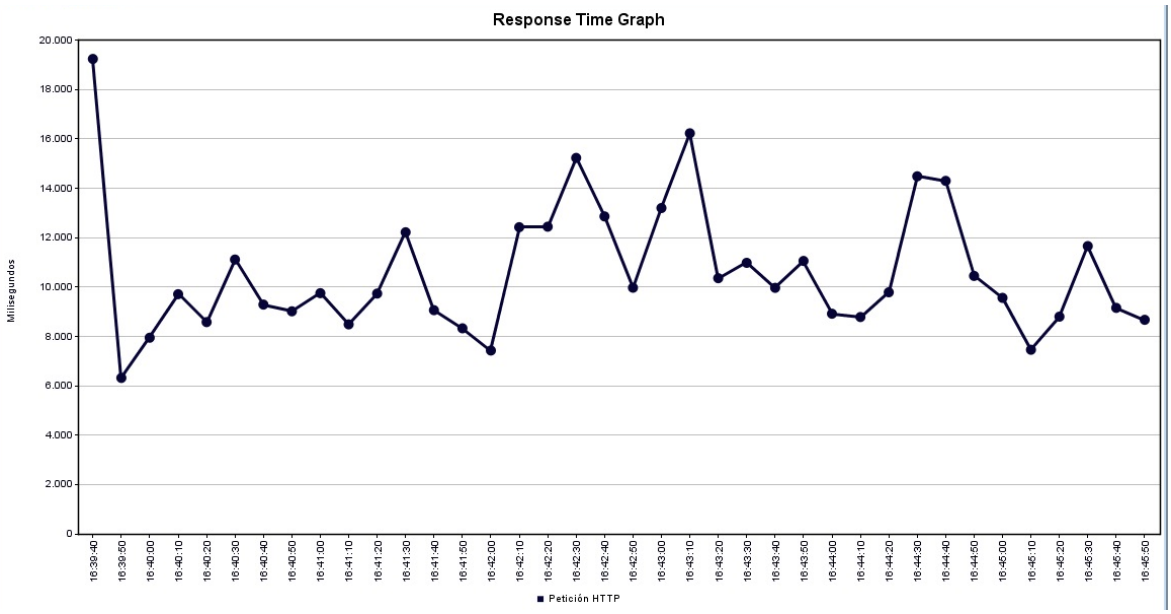


Figura 22 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 3 Fuente: Creación del software libre Jmeter como resultado de las pruebas

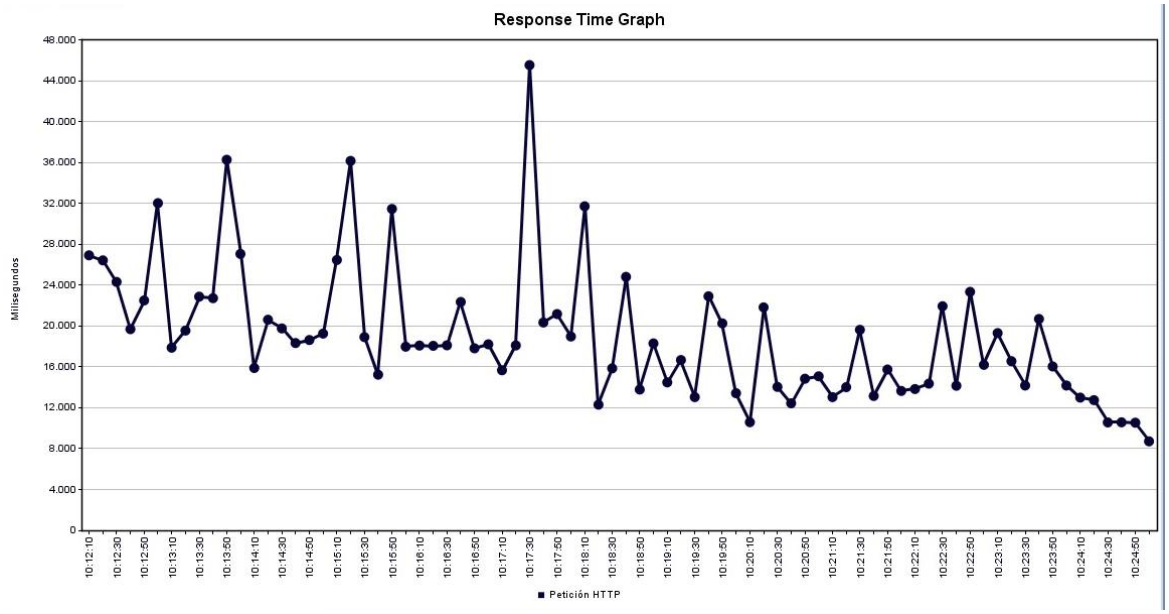


Figura 23 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 4 Fuente: Creación del software libre Jmeter como resultado de las pruebas

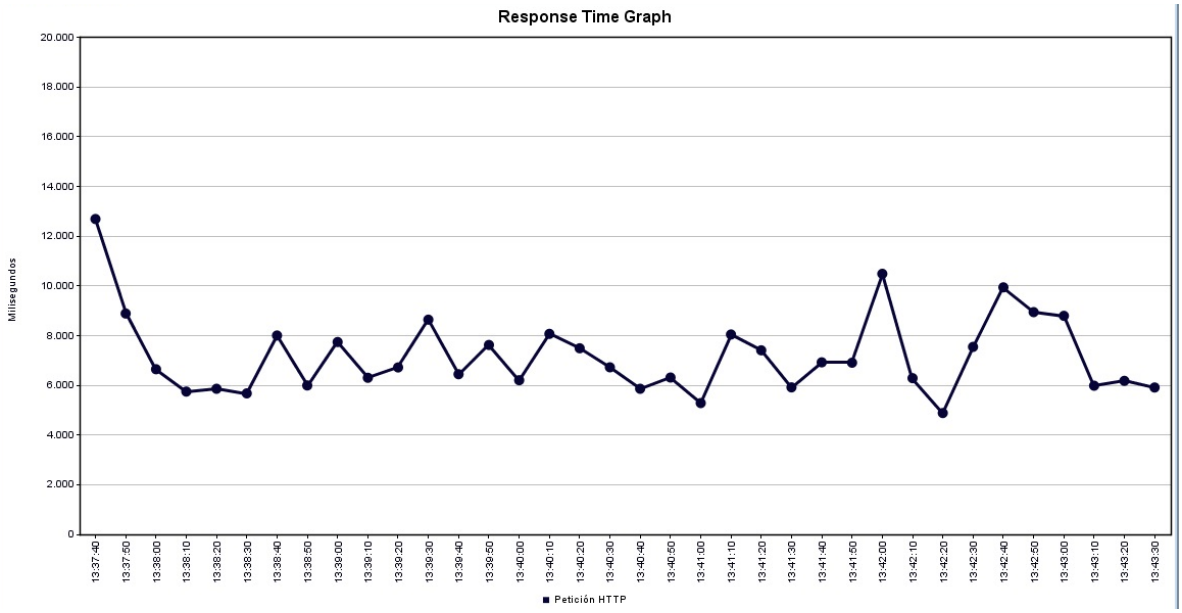


Figura 24 Gráfica de resultados pruebas de estrés servidor eltiempo.com día 5 Fuente: Creación del software libre Jmeter como resultado de las pruebas

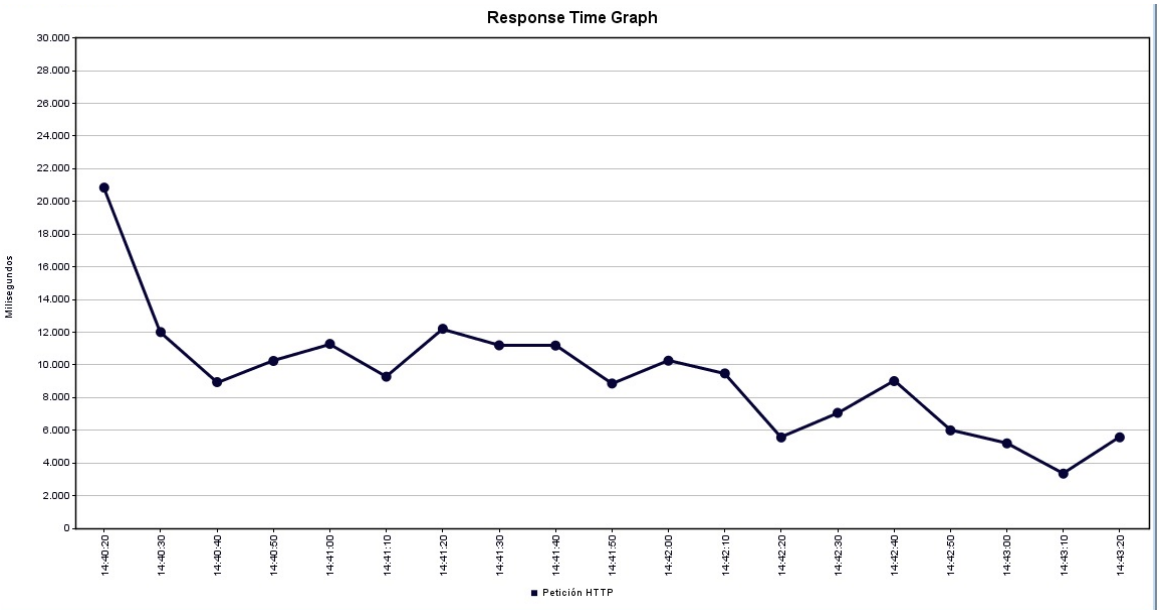


Figura 25 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 1 Fuente: Creación del software libre Jmeter como resultado de las pruebas

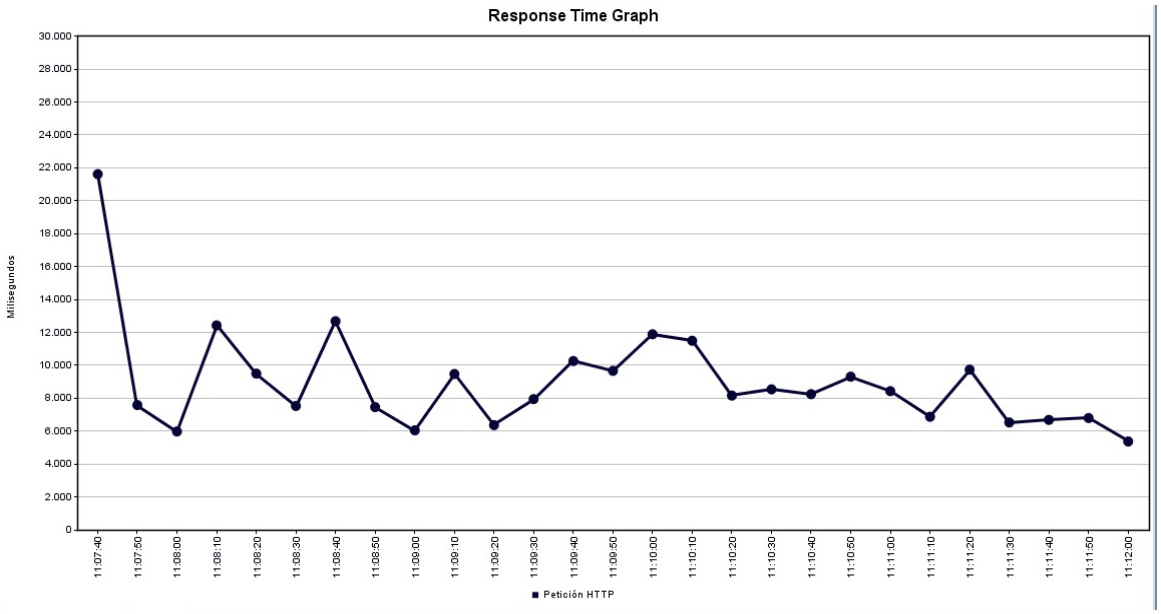


Figura 26 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 2 Fuente: Creación del software libre Jmeter como resultado de las pruebas

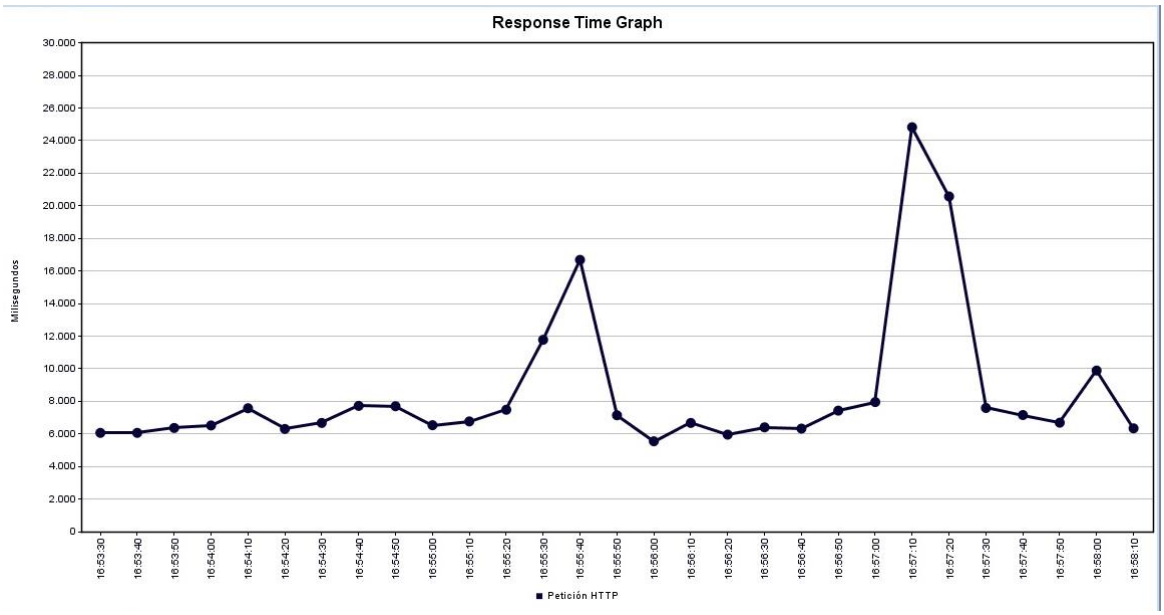


Figura 27 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 3 Fuente: Creación del software libre Jmeter como resultado de las pruebas

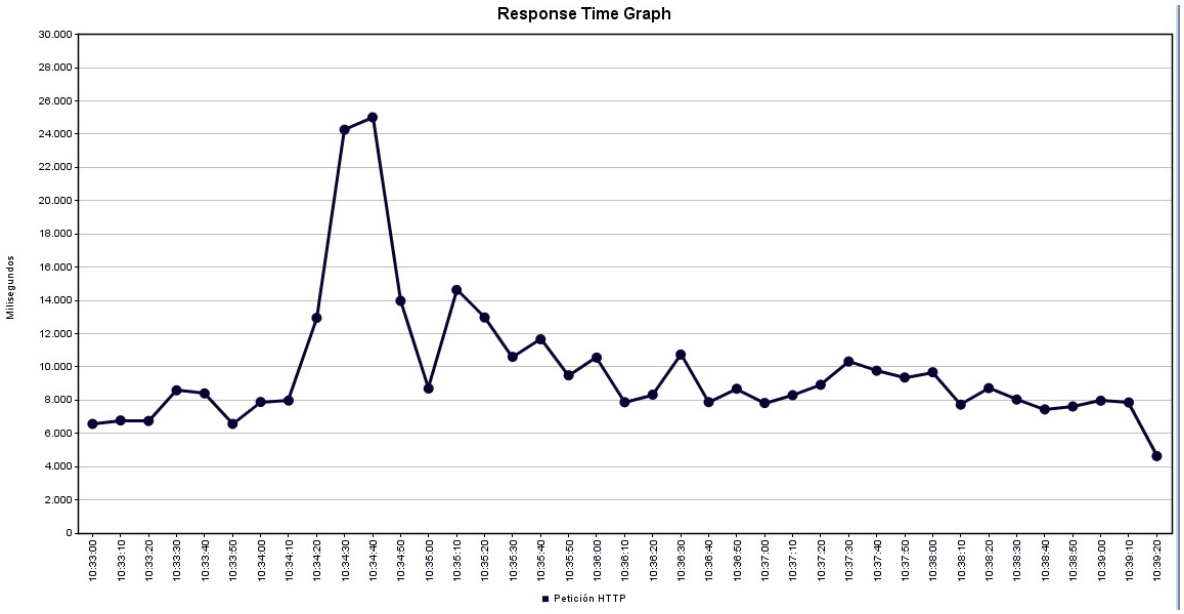


Figura 28 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 4 Fuente: Creación del software libre Jmeter como resultado de las pruebas

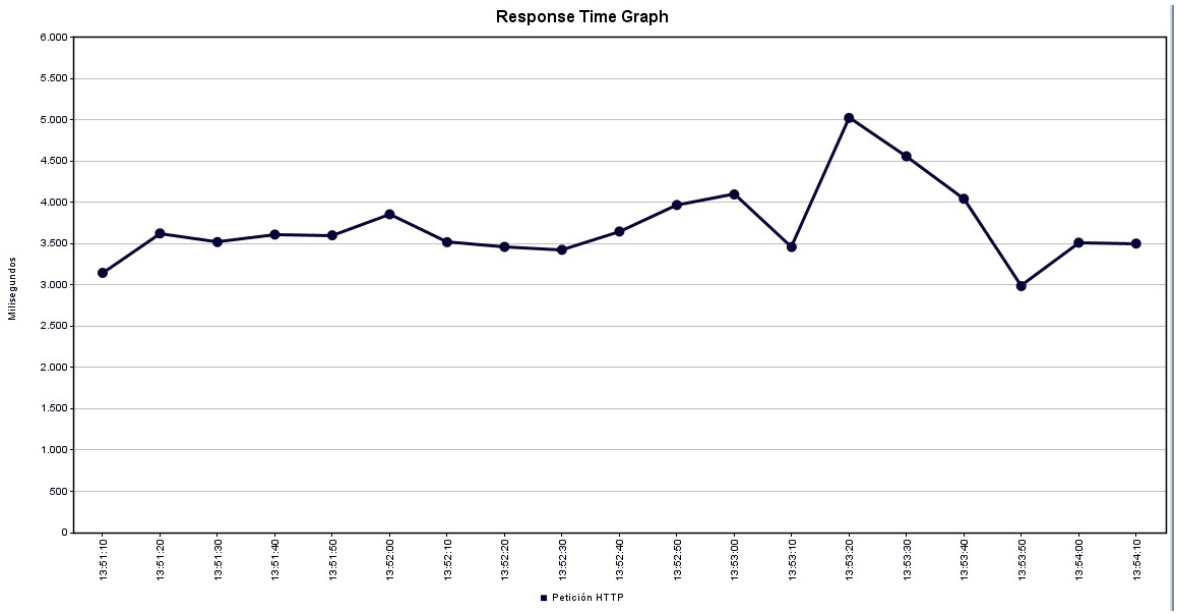


Figura 29 Gráfica de resultados pruebas de estrés servidor caracoltv.com día 5 Fuente: Creación del software libre Jmeter como resultado de las pruebas

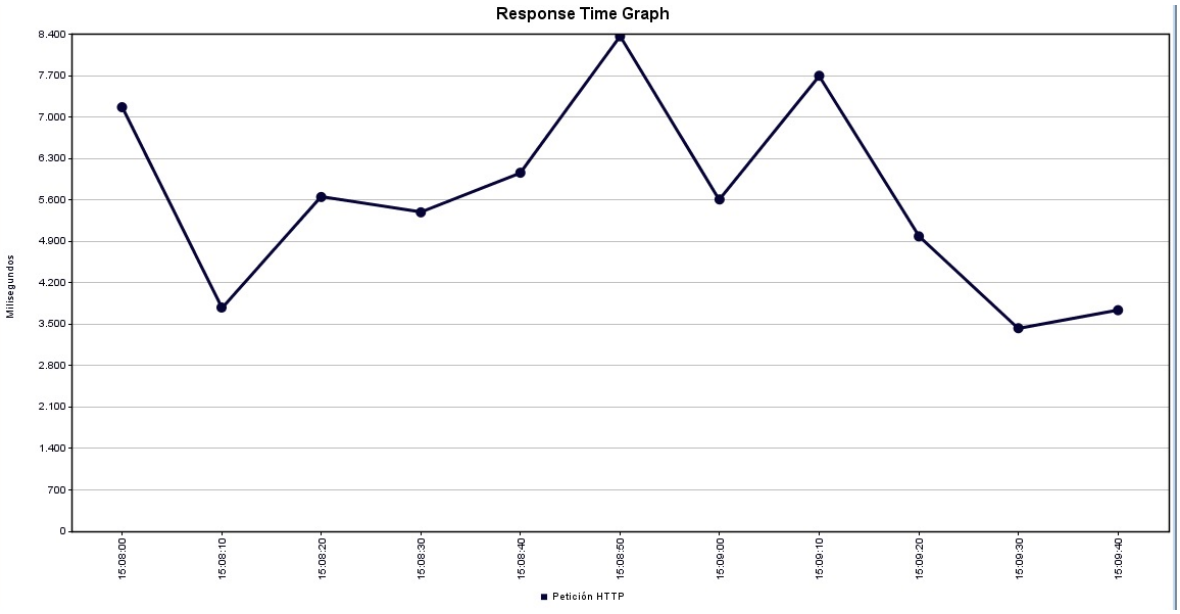


Figura 30 Gráfica de resultados pruebas de estrés servidor facebook.com día 1 Fuente: Creación del software libre Jmeter como resultado de las pruebas

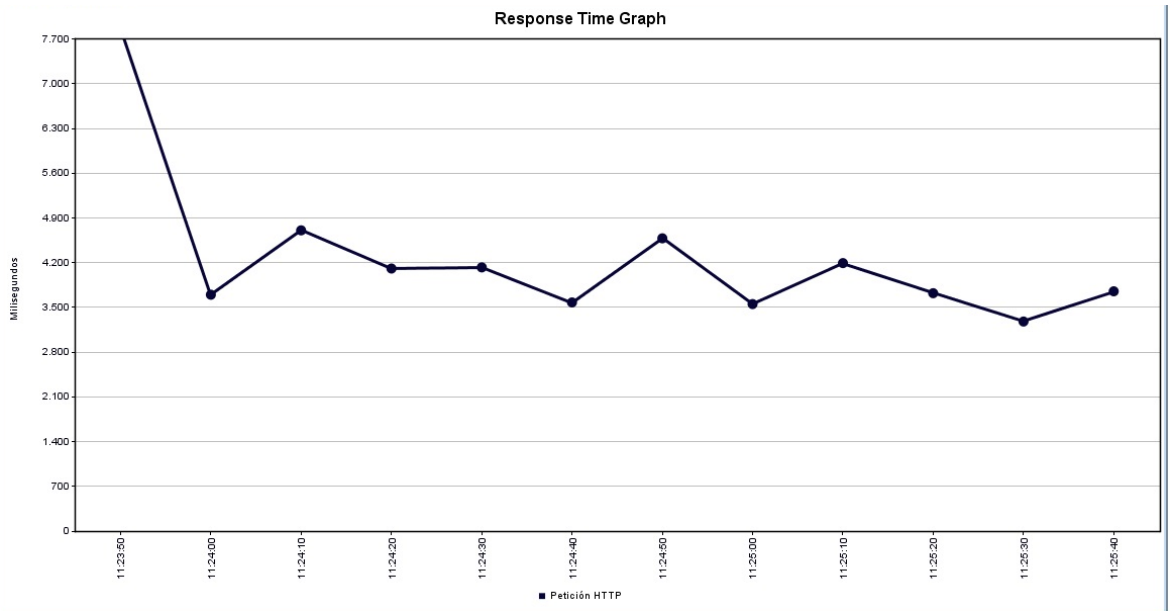


Figura 31 Gráfica de resultados pruebas de estrés servidor facebook.com día 2 Fuente: Creación del software libre Jmeter como resultado de las pruebas

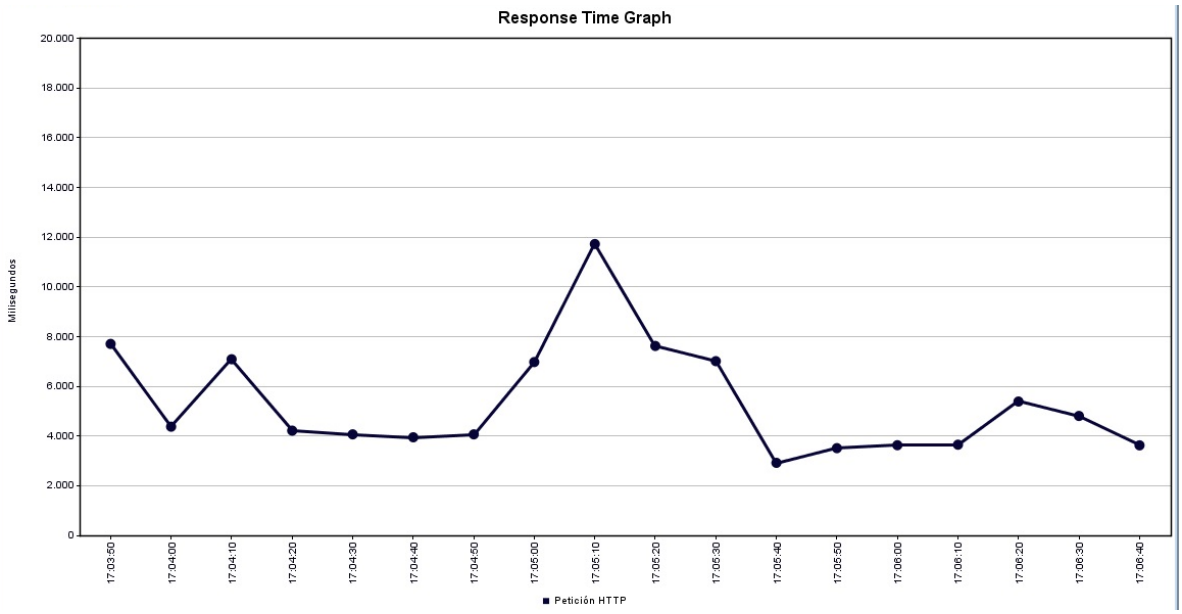


Figura 32 Gráfica de resultados pruebas de estrés servidor facebook.com día 3 Fuente: Creación del software libre Jmeter como resultado de las pruebas

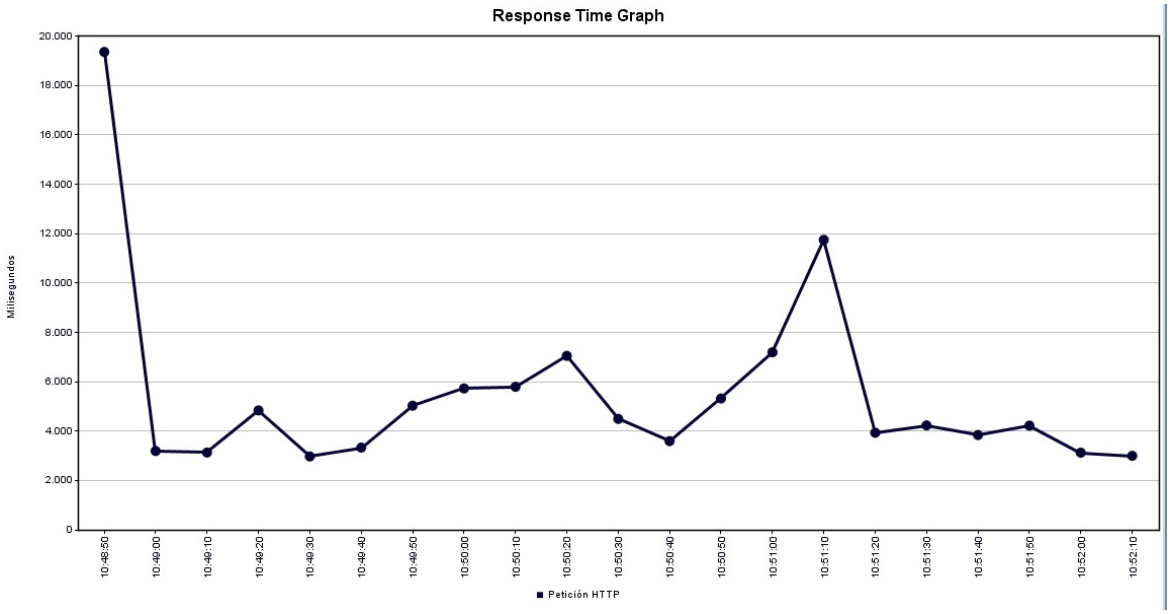


Figura 33 Gráfica de resultados pruebas de estrés servidor facebook.com día 4 Fuente: Creación del software libre Jmeter como resultado de las pruebas

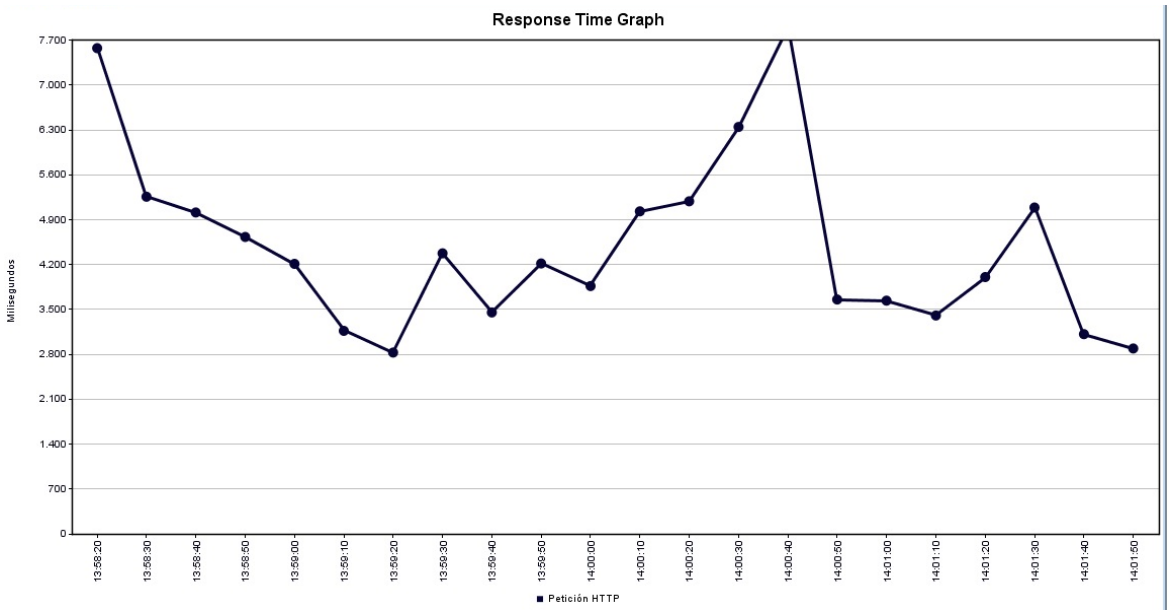


Figura 34 Gráfica de resultados pruebas de estrés servidor facebook.com día 5 Fuente: Creación del software libre Jmeter como resultado de las pruebas

En lo que respecta a Google, el error promedio fue de 2.792%, fue bajo y por lo tanto aceptable para un margen de entre 14 y 16 nodos de extremo a extremo, hay que tener en cuenta que es una página extremadamente sencilla sin contenidos multimedia y por lo tanto no requiere de mucho esfuerzo por parte ni del cliente ni del servidor por lo tanto el tráfico no demanda gran cantidad de recursos.

En cuanto a tiempos de respuesta durante los periodos de la prueba en el 40% de las pruebas la velocidad de los ACK bajó, un 40% se mantuvo estable y un 20% aumentó.

En lo que respecta a Youtube, el error promedio fue de 50,776%, claramente un margen de error bastante alto muy por encima de lo aceptado (Myers, B. A. (1985)), en la cantidad de nodos siempre fue constante siendo un total de 14, durante las pruebas se observó que la trasmisión de lo que respecta a imágenes, texto e hipervínculos se comportó en su mayoría bien la verdadera congestión se vio en el streaming, gran cantidad de paquetes que contienen información del video como tal no llegaba al destino y era necesario una retrasmisión lo que claramente genera más tráfico y por lo tanto más congestión.

En cuanto a tiempos de respuesta en ninguna de las pruebas se mostró estabilidad, los tiempos siempre variaron de manera errática independientemente del número de usuarios.

En lo que respecta a eltiempo.com, el error promedio fue de 5,088%, un error aceptable (Myers, B. A. (1985)), durante las pruebas se detectaron entre 10 y 15 nodos, se observó que el contenido de imágenes, hipervínculos y texto se comportó bien en su mayoría la congestión la ocasionaron principalmente la publicidad, la cual está comprendida por imágenes de tipo gif, por videos en streaming y archivos de sonido.

En cuanto a tiempos de respuesta durante el 100% de las pruebas se observó una respuesta constante aunque se presentó un pico en el aumento de los tiempos cuando se activaban los contenidos multimedia siendo el más crítico el de video.

En lo que respecta a caracoltv, el error promedio fue de 4,274%, un error aceptable, durante la totalidad de las pruebas el número de nodos fue constante siendo un total de 12, durante

las pruebas se observó que el contenido de imágenes, hipervínculos y texto se comportaron bien en su mayoría la congestión la ocasionaron principalmente la publicidad de sus contenidos y pequeños videos con avances de la programación.

En cuanto a tiempos de respuesta se pudo observar aumentaba en función de la cantidad de usuarios aun así para cada caso los tiempos se mantienen constantes de igual manera se observó un pico de aumento de tiempo cuando se activaban los contenidos multimedia siendo el más crítico el de video.

En lo que respecta a facebook, el error promedio fue de 3.4%, un error aceptable, durante la totalidad de las pruebas se detectaron un número constante de nodos siendo un total de 15, durante las pruebas se observó que el contenido de imágenes, hipervínculos y texto se comportaron bien en su mayoría, hubo una pequeña congestión durante los momentos de la carga de imágenes de álbumes.

En cuanto a tiempos de respuesta se pudo observar que no son constantes y aunque aumentan y disminuyen cuando se realiza carga de grandes cantidades de imágenes los tiempos erráticos también se presentan cuando no se realizan dichas cargas.

2.3.5 COMPARACIÓN DE RESULTADOS CON OTROS AUTORES

Durante el proceso investigativo se decidió comparar los resultados de estas pruebas con algunas similares de otros autores con el fin de determinar si existe una consistencia en los resultados, concretamente en el trabajo titulado: SEGURIDAD, RENDIMIENTO Y QoS EN TCP (Luque Calderón & Vela Porras, 2015) se puede observar según los resultados de las simulaciones que se realizaron que el nivel de error para el protocolo TCP es mayor para los casos de las trasmisiones streaming o multimedia.

En este trabajo se probaron 4 escenarios, que aunque si bien son más pequeños que las pruebas realizadas en este documento y con una menor cantidad de paquetes en estos escenarios se encontró un margen de error entre el 0.275% pasando por 0.425% y 0.592%

hasta llegar al máximo de 44.35% claramente el ultimo es el más crítico este hace referencia a QoS.

De igual manera en el trabajo titulado: PREDICCIÓN DE TRÁFICO A TRAVES DE REDES NEURONALES ARTIFICIALES (Luque Calderon & Vela Porras, 2016). En este trabajo se realizó principalmente una investigación para lograr como su nombre da a entender con ayuda de una red neuronal una predicción del tráfico total de una red con el fin de generar alertas tempranas para las congestiones.

	Escenario 1		**Escenario 2**		**Escenario 3**		**Escenario 4**	
Paquetes total	35146	100%	39697	100%	41898	100%	94199	100,00%
Paquetes promedio	66,53	-	66,27	-	69,95	-	158,32	-
TCP	35036	99,687%	39583	99,713%	41738	99,618%	90203	95,76%
Retrasmisión	105	0,299%	108	0,270%	107	0,258%	27090	28,76%
Duplicados	103	0,293%	2	0,005%	70	0,167%	14681	15,59%

Tabla 11 Resultados Simulaciones en 4 escenarios fuente: (Luque Calderón & Vela Porras, 2015)

De ese trabajo lo que se comparó fue el hecho de que de igual manera se manejó el streaming como aspecto que más generó tráfico tanto antes como después de la implementación de la red neuronal.

Usaron una simulación de una red mucho más pequeña de 3 nodos y enviando entre 50 mil y 51 mil paquetes logrando una mejora del 4% que la neurona aplica.

En conclusión se puede observar en los trabajos de otros autores que han realizado simulaciones sobre capacidad de carga del protocolo TCP e incluyendo el presente artículo todos coinciden en que la cantidad de congestión es proporcional a la cantidad de usuarios,

de nodos y del tipo de información que se está enviando siendo el más crítico el streaming o cualquier tipo de contenido multimedia.

2.3.6 CONCLUSIONES ESTADO DEL ARTE

Las páginas sin contenido multimedia, solo texto, botones, hipervínculos, etc presentan una baja cantidad de congestión.

El contenido multimedia en especial el video en streaming presenta la mayor congestión de todas, aproximadamente el 50% de los errores de trasmisión se deben a este factor, esto genera tiempos erráticos y numerosos errores independientes del otro contenido de la misma página.

Claramente es importante bajar los tiempos de respuesta al máximo o por lo menos lograr que aumenten en proporción a la cantidad de trasmisiones, variando los tamaños de segmentos conservando el mínimo de 12000 bytes (1,333 megabytes) hasta máximo 50000 bytes (5,55 megabytes) lo que es una reducción del 16% con respecto al actual, esto debido a que con 50000 bytes se observó un desempeño bueno.

Se debe tener en cuenta que páginas con varios aplicativos internos (a menudo invisibles para el usuario (chats)) presentan envíos de cantidades impredecibles de información lo que genera una congestión adicional al comportamiento visible por el usuario.

Una situación que se presenta comúnmente consiste en que la cantidad de errores es baja, pero los tiempos de respuesta son grandes llegando al orden de los segundos siendo el máximo el caso de youtube con 66874ms lo cual se traduce en contenidos cargados correctamente pero de manera muy lenta.

Finalmente apoyado en lo visto durante las pruebas se concluye que el actual tipo de contenido y servicios que internet presenta es necesario la implementación de una nueva generación de algoritmos de control de congestión enfocado en el contenido multimedia:

- Video
- Audio
- Aplicaciones interactivas (juegos)

De igual manera generar algoritmos enfocados a páginas que presente applets propios aparte de los servicios que brindan (chat), o actualizaciones de contenido automáticas.

Estos algoritmos también se deben concentrar en lograr disminuir los tiempos de respuesta de los ACK preferiblemente a 50000ms tiempos buenos para la cantidad de usuarios de las pruebas según lo observado durante las mismas, lo que significaría una disminución de aproximadamente el 30% del tiempo con el fin de mejorar la experiencia de los usuarios, lograr superar la barrera de las páginas que cargan correctamente pero de manera lenta.

3. DISEÑO DEL MODELO ALGORÍTMICO PARA LIMITAR LA CONGESTIÓN DEL PROTOCOLO TCP EN REDES DE EXTREMO A EXTREMO

Los principales retos que se identificaron durante la etapa investigativa son:

- La relación entre la capacidad disponible del buffer en cada nodo y la cantidad de paquetes atendidos de este.
- El tamaño más adecuado de los paquetes a enviar teniendo en cuenta que la capacidad disponible de los buffers es variante en el tiempo.
- Determinar la franja de tiempo en el que los ACK se comportan de una manera correcta y cuando se comportan de una manera que se puede interpretar como "congestión"
- A qué tipo de tráfico se le debe dar prioridad y bajo qué parámetros se debe hacer esto con el fin de limitar una congestión de red cuando esta suceda.
- Determinar la cantidad de tiempo en la cual el modelo debe reevaluar las características mencionadas con el fin de realizar los ajustes necesarios.

Para lograr determinar los valores para el modelo se realizó una investigación acompañada de pruebas en infraestructura real las cuales indicaron el comportamiento actual de las redes LAN-WAN-LAN bajo diferentes tipos de tráfico y servicios con lo cual se determinó lo siguiente.

3.1 BUFFER

Es conocido que la cantidad de nodos y por lo tanto de buffers dentro de una red LAN varia de una a otra y no es algo controlable, así que el factor a tener en cuenta para cada nodo corresponde a la capacidad del buffer (Bing, Wireless LANs and Home Networks, 2001).

Se tuvo en cuenta que dentro de cada red LAN cada paquete puede pasar por diferentes routers o switches y que en cada uno de ellos la capacidad varia, de igual manera se toma

en cuenta que el punto donde termina la LAN y empieza la WAN corresponde a otro cambio de velocidades, tamaños de segmentos y routers diferentes (Giambene, 2005):



Figura 35 Ejemplo de la diferencia de capacidades de buffers en routers de una misma LAN, Fuente: Creación propia con uso de imágenes del banco de imágenes gratuitas OpenClipArt

Recurriendo a la teoría de colas se tomó como referencia el modelo M/M/1/K, cuyas principales características son (Spiegel, 1975):

- Presenta un proceso de Poisson
- El tiempo de servicio de los paquetes es de tipo exponencial.
- Es una cola que presenta un único servidor.
- La cola (buffer) de dicho servidor tiene una capacidad finita de limite ***K***



Figura 36 Modelo M/M/1/K, fuente: creación propia con ayuda del software gratuito para procesado de imágenes GIMP

Según la notación para este sistema se tiene:

- La cantidad de paquetes = N
- El tiempo que permanece cada paquete en sistema = T (en segundos)
- Tasa de entrada de paquetes por unidad de tiempo = λ
- Tasa de paquetes que son procesados por unidad de tiempo = μ

Además del parámetro que más interés presenta para el presente trabajo, la intensidad del sistema denotada con la letra **ρ**

$$\boldsymbol{\rho = \frac{\lambda}{\mu}}$$

Ecuación 1 Intensidad del sistema en un modelo de colas M/M/1/K

Este parámetro es el que se traduce en congestión de la red, si la intensidad del sistema es menor que la totalidad de la capacidad del mismo se entiende que la capacidad de atender paquetes por unidad de tiempo es superior a la tasa de paquetes que llega por lo tanto el sistema opera sin contratiempos:

$$\boldsymbol{\rho < 1}$$

Ecuación 2 Intensidad del sistema en una cola M/M/1/K sin congestión

Si la intensidad del sistema es igual a la totalidad de la capacidad del mismo se entiende que la capacidad de atender paquetes por unidad de tiempo está al límite de la tasa de paquetes que llega, la cola está llena, pero se atienden la totalidad de paquetes por lo tanto el sistema está estable.

$$\boldsymbol{\rho = 1}$$

Ecuación 3 Intensidad del sistema en una cola M/M/1/K estable

Si la intensidad del sistema es mayor que la totalidad de la capacidad del mismo se entiende que la capacidad de atender paquetes por unidad de tiempo es inferior a la tasa de paquetes que llega el sistema presenta pérdidas de paquetes por sobredimensionamiento de solicitudes, el sistema está en congestión:

$$\boldsymbol{\rho > 1}$$

Ecuación 4 Intensidad del sistema en una cola M/M/1/K cuando se presenta congestión

Teniendo la teoría anteriormente expuesta en cuenta y apoyados de los datos obtenidos durante las pruebas y seguimientos realizadas en la red LAN-WAN-LAN se determinó que la magnitud de la capacidad del buffer (Bing, 2010) determinará el tamaño máximo de los

paquetes dividendo la capacidad de este (entendida como un 100%) en el total de servicios de mayor relevancia en la red, para este caso puntual se tomaron un total de 6 divisiones ya que según el seguimiento realizado ese es el número total de servicios que presentan la totalidad del trafico TCP, de esta manera se logra encontrar un equilibrio entre evitar la pérdida de paquetes (lo cual aumenta el nivel de congestión generando retrasmisiones) y el no aumentar en exceso la cantidad de retardo que cada paquete experimenta en la cola haciendo que este no deba esperar más de lo debido para ser procesado:

Porcentaje buffer libre (%)	Clasificación
Entre 100 y 90	Operación sin contratiempos
Entre 89 y 70	
Entre 69 y 50	Sistema estable
Entre 49 y 30	
Entre 29 y 10	Congestión
Entre 9 y 1	

Tabla 12 Capacidades de buffers definidas y su correspondiente clasificación, Fuente: Creación propia

3.2 TAMAÑO DE LOS PAQUETES

En lo que respecta al tamaño de los paquetes se tomó en cuenta, por una parte, la teoría en la que opera TCP, la cual indica que el tamaño máximo permitido es de 64KB (Briceño, 2005) eso indica el valor máximo que tendrían los paquetes, aun así, durante las pruebas y el seguimiento realizado se observó que el tamaño variaba dependiendo del tipo de servicio, el origen y/o destino de este, únicamente Google (la página más sencilla que se usó en las pruebas) arrojo tamaños máximos de entre 50KB y 60KB.

Durante esos días se logró observar que el porcentaje de error disminuía si el tamaño del paquete también lo hacía:

- A paquetes pequeños menor era la cantidad de errores de trasmisión
- A paquetes grandes mayor era la cantidad de errores de trasmisión

Aun así como ya se mencionó el tamaño variante de los paquetes en cada servicio es un factor que varía demasiado no tiene patrones o comportamientos definidos lo cual genera picos de trasmisiones (congestión) o periodos de operaciones tranquilas cambiantes entre si casi que de forma aleatoria.

Debido a lo mencionado en esta parte del proceso que se debió encontrar un punto de equilibrio entre el tamaño de los paquetes y la cantidad de buffer disponible, de tal manera que se lograra evitar la pérdida de paquetes sin que esto afectara la calidad del servicio.

Para esto se decide eliminar el factor de aleatoriedad de tamaño entre paquetes grandes y/o pequeños de manera simultánea y se decide que el modelo tendrá una serie de tamaños fijos para los paquetes con excepción del último paquete a trasmitir ya que este a menudo es más pequeño, de esta manera se logra mantener el sistema mucho más estable y controlado.

Ya habiendo mencionado esto y con el apoyo de los resultados de las pruebas y seguimientos en la red LAN-WAN-LAN se decidió tomar un rango mínimo de 1 KB hasta un valor máximo de 50KB (debido a que solo un porcentaje depreciable de los nodos superó los 50KB lo cual lo convierte en un tope máximo adecuado), esto se dividió en 6 secciones que corresponden a los 6 servicios más usados en lo que respecta a TCP las cuales

agrupan todo el rango previamente mencionado y las cuales son las que presentaron un mejor comportamiento durante las pruebas y seguimientos.

Tamaño del Paquete (KB)	Clasificación
1	Paquetes Pequeños
10	
20	Paquetes Promedio
30	
40	Paquetes Grandes
50	

Tabla 13 Tamaños de los paquetes a trasmitir y su correspondiente clasificación, Fuente: Creación propia

3.3 RELACIÓN ENTRE TAMAÑO DEL BUFFER Y EL TAMAÑO DE LOS PAQUETES

Una vez se realizaron las clasificaciones de la cantidad de buffer disponible y del tamaño del paquete se procede a relacionarlos con la intención de encontrar un equilibrio entre evitar la pérdida de paquetes, (lo cual aumenta el nivel de congestión generando retrasmisiones) y no aumentar en exceso la cantidad de retardo que cada paquete experimenta en la cola (lo que hace que este deba esperar más de lo debido para ser procesado).

Para lograr este equilibrio se relaciona una proporción directa en cuanto a las magnitudes de estos dos parámetros:

- Entre más buffer disponible se tenga mayor será el tamaño de los paquetes: Esto debido a que el buffer al tener una cantidad grande de espacio indica que tiene pocos paquetes en cola y que puede recibir paquetes con un tamaño más grande, lo cual acelera el proceso de la comunicación, por ejemplo: Entre más rápido se trasmita toda la información (paquetes necesarios) entre dos extremos más rápido se concluye el servicio y se libera ese canal.
- Entre menos buffer disponible se tenga, menor será el tamaño de los paquetes: Esto debido a que el buffer al tener poca cantidad de espacio es propenso a rechazar paquetes, los cuales luego necesitarán retrasmisión, al recudir el tamaño se reduce la cantidad de paquetes que serán rechazados.

Finalmente, la relación queda de la siguiente manera:

Porcentaje buffer libre (%)	Tamaño del Paquete (KB)
Entre 100 y 90	50
Entre 89 y 70	40
Entre 69 y 50	30
Entre 49 y 30	20
Entre 29 y 10	10
Entre 9 y 1	1

Tabla 14 Relación entre la cantidad de buffer disponible y el tamaño de los paquetes, Fuente: Creación propia

3.4 TIEMPO DE LOS ACK

El factor del tiempo de los ACK es muy importante debido a que este determina la velocidad con la que los paquetes están siendo recibidos y procesados para esta estimación se

calculó el promedio de promedios de estos tiempos de los seguimientos realizados durante 5 días o como es conocido matemáticamente: la gran media (V.E, 1975):

Promedio día 1 LAN:

$$T_{dia1\ LAN}(ms) = 2$$

Ecuación 5 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 1 Bogotá-Medellín en milisegundos

Promedio día 2 LAN:

$$T_{dia2\ LAN}(ms) = 2.2$$

Ecuación 6 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 2 Bogotá-Medellín en milisegundos

Promedio día 3 LAN:

$$T_{dia3\ LAN}(ms) = 1.3$$

Ecuación 7 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 3 Bogotá-Medellín en milisegundos

Promedio día 4:

$$T_{dia4\ LAN}(ms) = 1.1$$

Ecuación 8 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 4 Bogotá-Medellín en milisegundos

Promedio día 5:

$$T_{dia5\ LAN}(ms) = 0.8$$

Ecuación 9 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 5 Bogotá-Medellín en milisegundos

Promedio de promedios (gran media) del total de ACK para la red LAN:

$$T_{ACK\ LAN}(ms) = \frac{2 + 2.2 + 1.3 + 1.1 + 0.8}{5} = 1.48 \approx 1.5ms$$

Ecuación 10 Gran media de los tiempos de los ACK de la red LAN

Promedio día 1 WAN:

$$T_{dia1\ WAN}(ms) = 5.1$$

Ecuación 11 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 1 Bogotá-Medellín en milisegundos

Promedio día 2 WAN:

$$T_{dia2\ WAN}(ms) = 4.8$$

Ecuación 12 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 2 Bogotá-Medellín en milisegundos

Promedio día 3 WAN:

$$T_{dia3\ WAN}(ms) = 4.6$$

Ecuación 13 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 3 Bogotá-Medellín en milisegundos

Promedio día 4 WAN:

$$T_{dia4\ WAN}(ms) = 5.4$$

Ecuación 14 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 4 Bogotá-Medellín en milisegundos

Promedio día 5 WAN:

$$T_{dia5\ WAN}(ms) = 5.3$$

Ecuación 15 Tiempo promedio de los ACK del seguimiento realizado a la red LAN-WAN-LAN día 5 Bogotá-Medellín en milisegundos

Promedio de promedios (gran media) del total de ACK para la red WAN:

$$T_{ACK\ WAN}(ms) = \frac{5.1 + 4.8 + 4.6 + 5.4 + 5.3}{5} = 5.04 \approx 5ms$$

Ecuación 16 Gran media de los tiempos de los ACK de la red WAN

Tal como se puede observar el error promedio de los ACK para la LAN fue de 1.5ms y de la WAN de 5ms valores que se tomaron como tope máximo para el modelo en lo que respecta a los tiempos de los ACK.

3.5 TIPOS DE SERVICIOS

Una vez realizado el cálculo del tope máximo de los tiempos de los ACK expuesto en la sección anterior se dispone a realizar la distribución de los servicios según el comportamiento de los factores anteriormente mencionados.

Para este caso en particular se van a manejar múltiples calidades de servicio (QoS) debido a que no todos los servicios presentan el mismo nivel de prioridad, según los datos obtenidos después de 5 días de seguimientos en una red LAN-WAN-LAN entre Bogotá y Medellín se encontró que ciertos servicios presentan más concurrencia que otros encontrando:

Servicio	Puerto	Uso (%)
FTP	20,21	95%
TelNet	23	
SMTP	25	
POP3	110	
DNS	53	
HTTP	80,8080	
UDP	69,161	
Restantes		5%
Total		100%

Tabla 15 Porcentajes promedio de seguimientos de diversos servicios en redes LAN-WAN-LAN

Es por esto que se decidió dividir en 7 grupos de servicios, 6 que corresponden a servicios sobre TCP y uno restante que corresponde a servicios UPD, (si bien el objetivo principal de la presente investigación es centrarse en TCP no se puede ignorar el tráfico UPD ya que también hace uso de los recursos de la red), filtrados según los tiempos de los ACK, los cuales son explicados en la siguiente sección.

3.6 RELACIÓN ENTRE TIEMPOS DE ACK Y LOS SERVICIOS A PRESTAR

Una vez se tiene en cuenta los tiempos máximos de los ACK siendo el tope máximo de 1.5ms para red LAN y 5ms para red WAN según lo calculado en la sección: 3.4 Tiempo de los ACK y conociendo los servicios que presentan una mayor concurrencia se relacionaron y se clasificaron en 6 grupos expuestos a continuación:

Tiempo de los ACK (Segundos)	Puertos habilitados
Tack >= 1.5ms	80,8080 (HTTP)
Tack <1.5ms y >= 1.29ms	53 (DNS), 80,8080 (HTTP)
Tack <1.29ms y >= 1.08ms	25 (SMTP), 110 (POP3), 53 (DNS), 80, 8080 (HTTP)
Tack <1.08ms y >= 0.87ms	20, 21 (FTP), 25 (SMTP), 110 (POP3), 52 (DNS), 80,8080 (HTTP)
Tack <0.87ms y >= 0.66ms	23 (Telnet), 20,21 (FTP), 25 (SMTP), 110 (POP3), 52 (DNS), 80,8080 (HTTP)
Tack <0.66ms y >= 0.45ms	***69, 161 (UDP)***, 23 (Telnet), 20,21 (FTP), 25 (SMTP), 110 (POP3), 52 (DNS), 80,8080 (HTTP)
Tack <0.45ms	Todos los puertos

Tabla 16 Relación entre los servicios a habilitar y el tiempo de los ACK para la red LAN

Tiempo de los ACK (Segundos)	Puertos habilitados
Tack >= 5ms	80,8080 (HTTP)
Tack <5ms y >= 4.29ms	53 (DNS), 80,8080 (HTTP)
Tack <4.29ms y >= 3.58ms	25 (SMTP), 110 (POP3), 53 (DNS), 80, 8080 (HTTP)
Tack <3.58ms y >= 2.87ms	20, 21 (FTP), 25 (SMTP), 110 (POP3), 52 (DNS), 80,8080 (HTTP)
Tack <2.87ms y >= 2.16ms	23 (Telnet), 20,21 (FTP), 25 (SMTP), 110 (POP3), 52 (DNS), 80,8080 (HTTP)
Tack <2.16ms y >= 1.45ms	***69, 161 (UDP)***, 23 (Telnet), 20,21 (FTP), 25 (SMTP), 110 (POP3), 52 (DNS), 80,8080 (HTTP)
Tack <1.45ms	Todos los puertos

Tabla 17 Relación entre los servicios a habilitar y el tiempo de los ACK para la red WAN

3.7 MODELO ALGORITMO PARA LIMITAR LA CONGESTIÓN DEL PROTOCOLO TCP EN REDES DE EXTREMO A EXTREMO

Una vez se tienen todas las métricas establecidas se procede a generar una serie de pasos que constituyen el modelo:

1. Determinar la cantidad de buffer dedicado para la red
2. Determinar la cantidad promedio de usuarios concurrentes
3. Dividir la cantidad de buffer libre entre el número de usuarios concurrentes
4. Dividir la totalidad del buffer entre el número de servicios más usados sobre TCP para así obtener un rango de control
5. Determinar los grupos de servicios más usados en la red y la prioridad que cada uno de estos presenta
6. Dividir el rango de 50KB como tamaño máximo del paquete hasta 1KB como tamaño mínimo del paquete entre el total de servicios más usados sobre TCP
7. Determinar el tiempo promedio de los ACK tanto en LAN como en WAN
8. Tomar el promedio como rango máximo de los tiempos y dividir este rango en la totalidad de los servicios más usados sobre TCP
9. Determinar el ancho de banda promedio por usuario para la red LAN y el ancho de banda promedio por enlace para la red WAN
10. Tomar ese valor promedio y dividirlo entre el total de servicios más usados sobre TCP
11. Relacionar los datos obtenidos del paso 4 con los obtenidos en el paso 6
12. Relacionar los datos obtenidos del paso 8 con los obtenidos en el paso 10

Finalmente, después de analizar detenidamente las métricas de buffer, tamaño del paquete, tiempos de los ACK, y servicios, y de expresar lo pasos a seguir se unifica todo en un solo modelo que es expone a continuación:

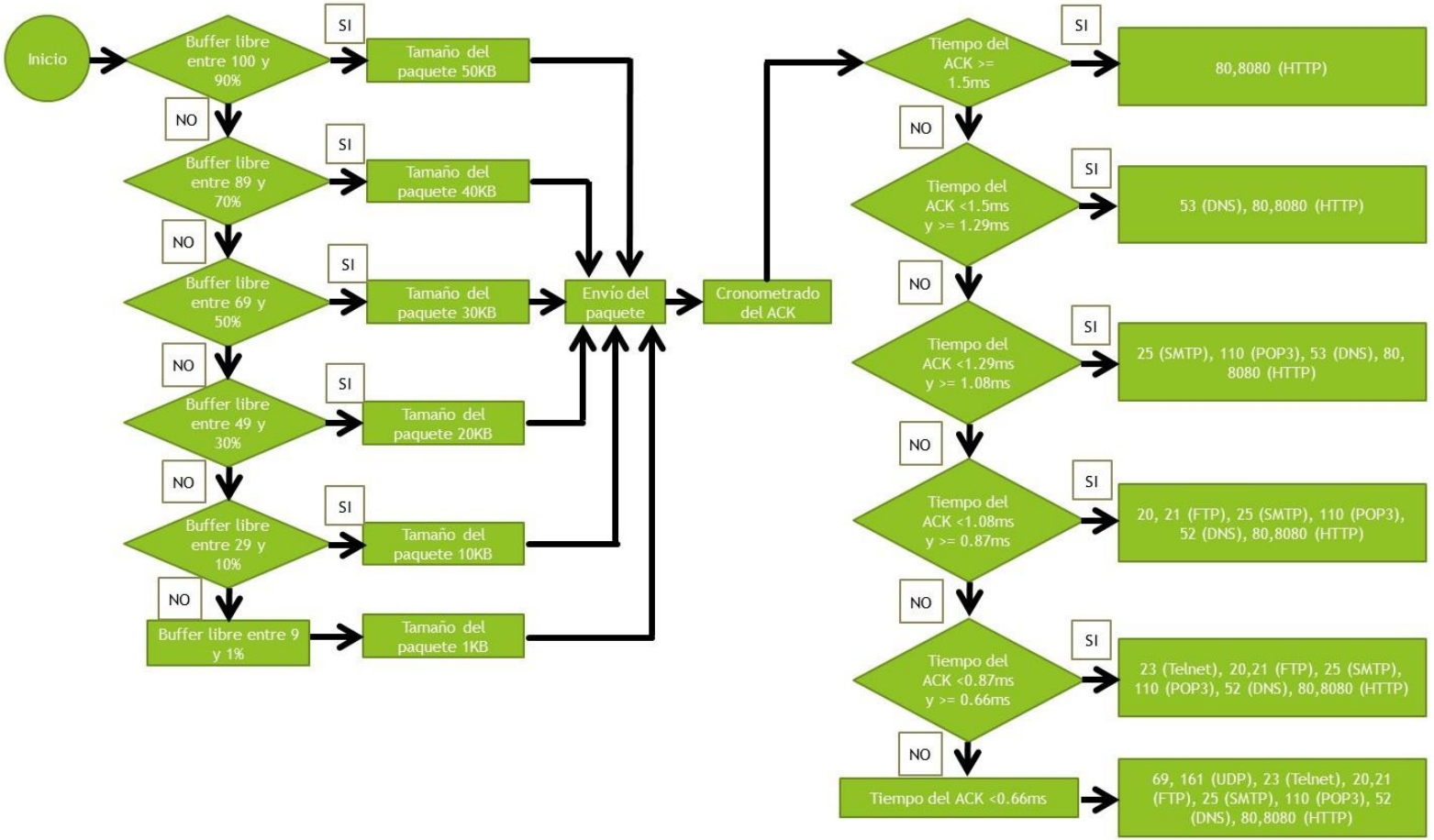


Figura 37 Modelo algorítmico para limitar la congestión del protocolo TCP en redes de extremo a extremo

4. HERRAMIENTAS DE SIMULACIÓN

Como parte del proceso de investigación se indagaron varias opciones de software para la simulación, a continuación, se procede a mostrar una selección de las herramientas encontradas:

4.1 NETWORK SIMULATOR 3 (NS3)

Es la tercera versión de un simulador de redes de eventos discretos, es un software de distribución gratuita bajo la licencia GNU GPLV2 el núcleo de su operación está en lenguaje C++ y las instrucciones se generan en Python (Azana Hafizah Mohd Aman, 2016).

Es ampliamente usado para simular como con anterioridad se mencionó redes de internet con la ventaja de soportar simulaciones de Wi-Fi, Wimax y LTE (nsman, 2017).

Permite el poder realizar emulaciones de los diferentes equipos que se usan en una red.

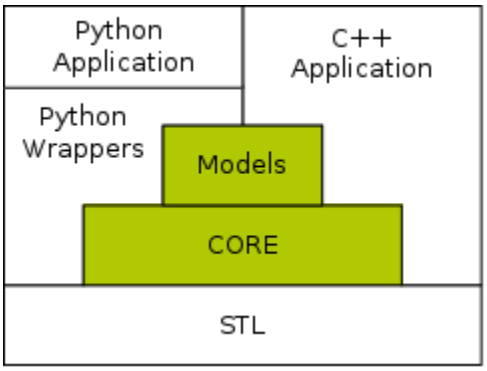


Figura 38 Estructura en diagrama de bloques simulador ns3, Fuente: https://www.nsnam.org/overview/key-technologies/ (consultado 04/04/2017)

4.2 QUALNET

Este es un software para simulación de protocolos fuertemente usado en simulaciones inalámbricas aunque también presenta las librerías correspondientes a redes cableadas las

instrucciones se ingresan mediante PARSEC el cual determina la cantidad de eventos discretos (Azana Hafizah Mohd Aman, 2016).

Permite una interfaz gráfica tridimensional en donde se puede planear, probar y monitorear actividades e diferentes redes según sea la necesidad (scalable-networks, 2017).

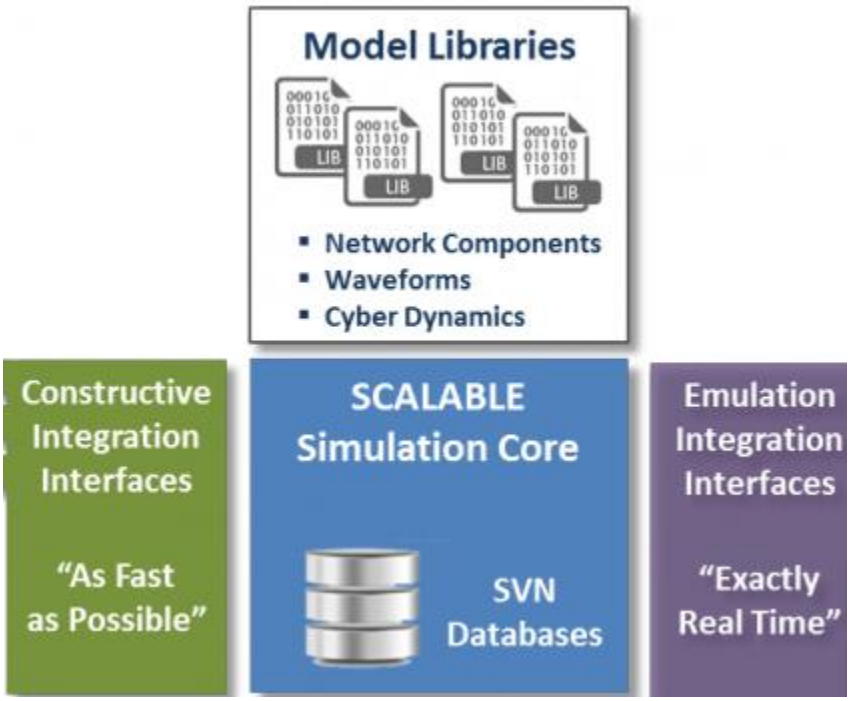


Figura 39 Estructura en diagrama de bloques simulador QUALNET, Fuente: http://web.scalable-networks.com/simulation-platform-architecture (consultado 04/04/2017)

4.3 JIST

De sus siglas en inglés: "Java en Tiempo de simulación " es una herramienta de software que permite la implementación de simulaciones de redes en Java, se utiliza mayormente a la vez con SWANS1.

En este software se pueden encontrar diferentes tipos de objetos que representan los elementos de red tales como los nodos, la simulación se forma por invocaciones de métodos entre estos objetos (Elias Weingartner, 2009).

Para lograr ejecutar las simulaciones, JiST carga las clases de manera personalizada que reescribe dinámicamente el código (jist, 2017).

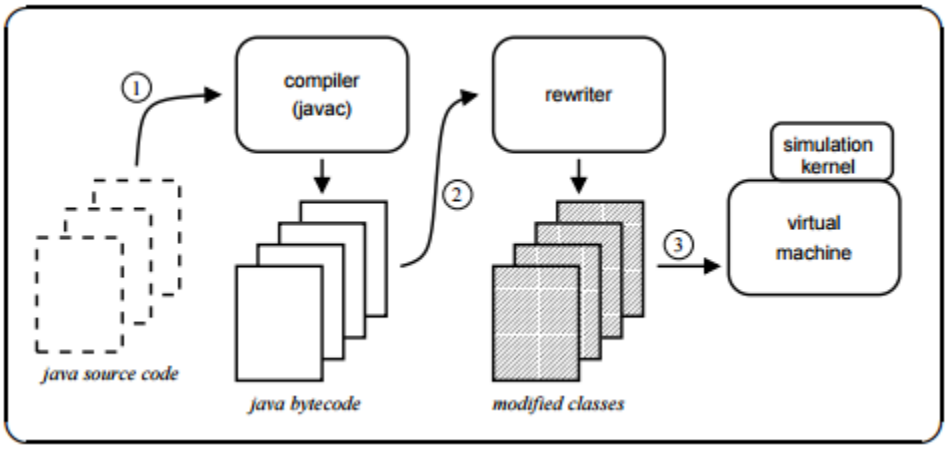


Figura 40 Estructura en diagrama de bloques simulador JiST, Fuente: http://jist.ece.cornell.edu/docs/040319-jist-user.pdf (consultado 04/04/2017)

4.4 OMNET++

Es un simulador fabricado por módulos y basado en objetos, está escrito en C++ y permite realizar simulaciones en tiempo real de redes de telecomunicaciones, permite la integración con bases de datos, además de que soporta otros lenguajes de programación como C# o Java (Shivangi Surati, 2016).

También presenta utilidades en simulación de sistemas de TI, teoría aplicada de colas, diversas arquitecturas de hardware y procesos de negocio.

También presenta una serie de modelos que presentan protocolos P2P. También soporta Java pero en este caso es necesario el uso de la extensión JSimpleModule (omnetpp, 2017).

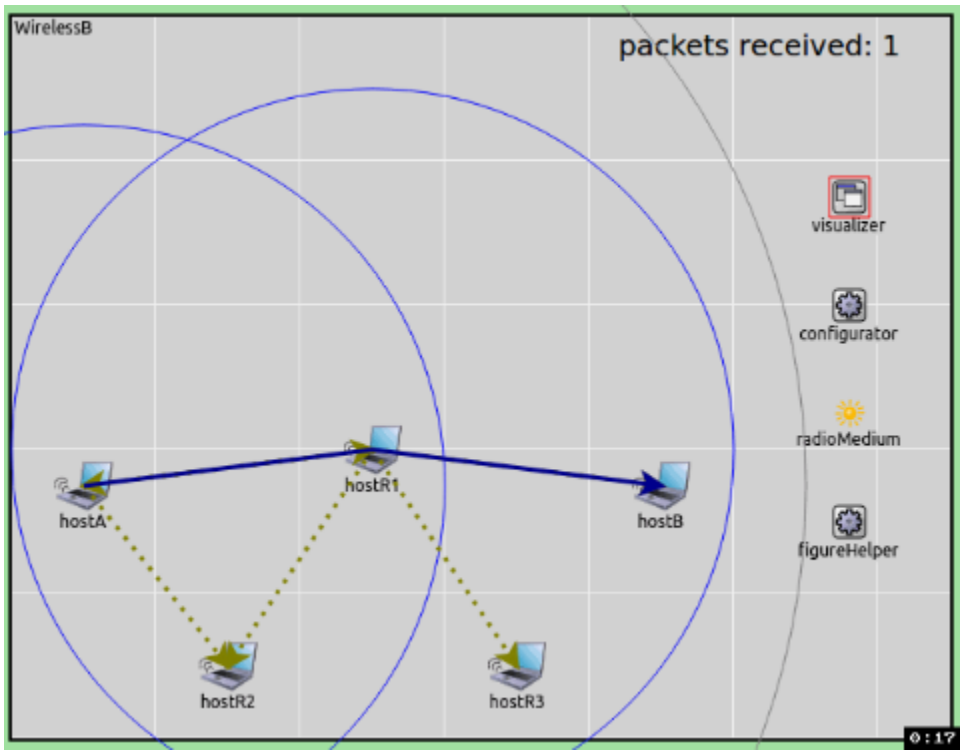


Figura 41 Ejemplo de simulación en OMNET++ fuente: https://omnetpp.org/ (consultado 04/04/2017)

4.5 NETWORK SIMULATOR 2 (NS2)

Una vez se ha desarrollado el modelo se procede a idear la manera de probarlo con el fin de determinar si cumple con lo esperado, para esto se hacen pruebas de estrés de las redes en simuladores desarrollados para esto, para este caso se eligió el simulador ns2 (Network Simulator 2) el cual es un simulador de eventos discretos (isi.edu, 2017) sobre redes LAN, WAN, MAN, Wi-Fi el cual permite el uso e implementación de diferentes protocolos, teniendo en cuenta el enfoque de este trabajo de investigación sobre redes LAN-WAN-LAN que usan protocolo TCP (Shivangi Surati, 2016).

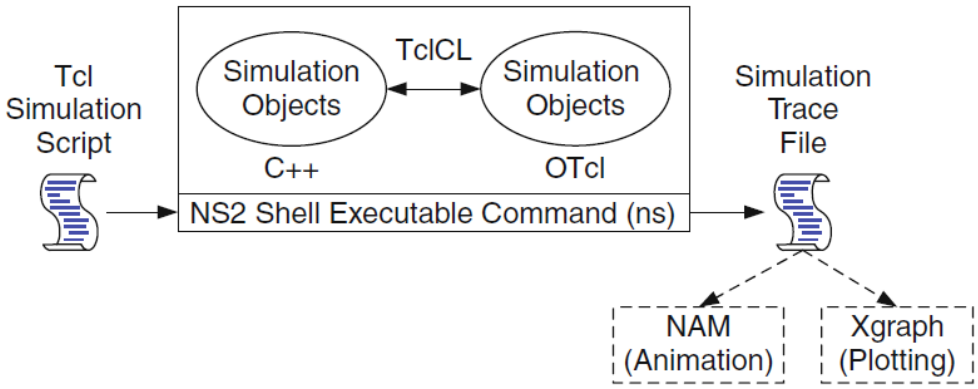


Figura 42 Estructura en diagrama de bloques simulador ns2, Fuente: https://ns2projects.org/ (Consultado 04/04/2017)

5. DISEÑO E IMPLEMENTACIÓN DE PRUEBAS

5.1 DISEÑO DE PROTOCOLO DE PRUEBAS

El protocolo para estas simulaciones consta de varios pasos:

- **Obtención de parámetros reales:**

  Se realizan mediciones de la red real sobre la cual se desea implementar el modelo algorítmico de control de congestión con el fin de determinar los valores de las métricas a tener en cuenta.

  Para el caso de este trabajo de investigación se realizó un seguimiento de 5 días sobre la red LAN-WAN-LAN que conecta a Bogotá con el departamento de Antioquia más exactamente la ciudad de Medellín.

- **Análisis de las métricas obtenidas:**

  Una vez se han obtenido las métricas producto de las mediciones mencionadas en el paso anterior se procede a analizarlas con el fin de determinar los datos máximos, rangos y total de valores que el modelo solicita.

  Para el caso de este trabajo de investigación se analizaron las métricas obtenidas utilizando sus promedios y la gran media (V.E, 1975) *de cada uno así obteniendo la cantidad de buffer libre, la cantidad de usuarios concurrentes, la cantidad proporcional de buffer para cada usuario concurrente, la cantidad de servicios esenciales, los tiempos máximos de ACK y su distribución según la cantidad de servicios, el ancho de banda proporcional para cada uno de ellos*.

- **Simulación de la red según las mediciones obtenidas**

  Se realiza una simulación de la red con los datos que fueron obtenidos durante su seguimiento, esto con el fin de obtener una comparación entre esos datos reales y los obtenidos mediante la simulación de esta red, esto con el objetivo de obtener dos variables de control con las cuales se compara el desempeño del modelo algorítmico para el control de congestión.

Para el caso de este trabajo de investigación se obtuvo una primera variable de control la cual corresponde a los datos de las mediciones de los seguimientos de 5 días sobre la red LAN-WAN-LAN y una segunda variable de control la cual se obtuvo de simular dicha red.

- **Simulación del total de escenarios**

  Teniendo en cuenta que el modelo maneja y relaciona varias métricas se debe tomar en cuenta el total de escenarios que se puedan producir según el número de rangos y divisiones que se generen y así simularlos en su totalidad con el fin de ver el comportamiento del modelo algorítmico para el control de congestión en cada uno de ellos.

  Para el caso de este trabajo de investigación se obtuvieron un total de 36 escenarios según la cantidad de rangos y divisiones obtenidas, se simularon los 36 con el fin de observar el desempeño del modelo en cada uno de ellos.

- **Análisis de resultados**

  Una vez se han concluido las simulaciones se procede a analizar los resultados de las mismas con la intención de determinar si el modelo realmente redujo la congestión y en caso de hacerlo en qué proporción con respecto a las variables de control.

  Para el caso de este trabajo de investigación se realizó el análisis de los 36 escenarios comparando el desempeño de cada uno con las variables de control así como un análisis final de los datos.

5.2 OBTENCIÓN DE PARÁMETROS REALES

Para la obtención de los datos reales se solicitaron sobre una red LAN-WAN-LAN de una red corporativa entre la cuidad de Bogotá y la ciudad de Medellín, en cada extremo la red LAN es propiedad del SENA y la red WAN es propiedad de Telefónica.

Estos datos corresponden a 5 días de seguimiento (entre el 27 y 31 de Marzo de 2017) de los cuales se obtuvieron los siguientes datos:

Routers LAN: Cisco NCS 5000 con 16 MB de buffer (Cisco, 2017)

Routers WAN: Cisco 910 con buffer de 1GB, pero solo disponible 500MB para la red monitoreada (Cisco, Cisco.com WAN, 2017)

El ancho de banda máximo parametrizado para cada usuario es de 6MB/s para la red LAN.

El ancho de banda máximo parametrizado para cada enlace es de 80MB/s para la red WAN.

Los puertos más usados son 7 los cuales corresponden al 95% del tráfico y son:

- 20, 21 ***FTP***
- 23 ***TelNet***
- 25 ***SMTP***
- 110 ***POP3***
- 53 ***DNS***
- 80, 8080 ***HTTP***
- 69, 161 ***UDP***

Porcentaje promedio de error durante los 5 días
0,085%

Tabla 18 Porcentaje de error de los 5 días de seguimiento en la red LAN-WAN-LAN entre Bogotá y Medellín

	Antioquia						
	Buffer LAN	Buffer WAN	Tamaño Promedio de los paquetes en LAN y WAN	Tiempo ACK en la LAN	Tiempo ACK en la WAN	Ancho de Banda por usuario por LAN	Ancho de Banda por canal WA
Día 1	16Megas/25 Usuarios concurrentes	500Megas/88 Usuarios Concurrentes	64Kb	2 Milisegundos	5,1 Milisegundos	4,5 Mb	18,4 Mb
Día 2	16Megas/22 Usuarios concurrentes	500Megas/90 Usuarios Concurrentes	64Kb	2,2 Milisegundos	4,8 Milisegundos	3,7 Mb	19,8 Mb

Día 3	16Megas/24 Usuarios concurrentes	500Megas/75 Usuarios Concurrentes	64Kb	1,3 Milisegundos	4,6 Milisegundos	4,1 Mb	17,6 Mb
Día 4	16Megas/27 Usuarios concurrentes	500Megas/87 Usuarios Concurrentes	64Kb	1,1 Milisegundos	5,4 Milisegundos	3,8 Mb	21,2 Mb
Día 5	16Megas/26 Usuarios concurrentes	500Megas/89 Usuarios Concurrentes	64Kb	0,8 Milisegundos	5,3 Milisegundos	4,3 Mb	22,5 Mb

Tabla 19 Resumen de las métricas obtenidas de los 5 días de seguimiento de la red LAN-WAN-LAN entre Bogotá y Medellín los días 27 al 31 de Marzo

5.3 ANÁLISIS DE LAS MÉTRICAS OBTENIDAS

Se realizó un análisis de los datos que se obtuvieron de la red LAN-WAN-LAN entre Bogotá y Medellín, se calcularon la gran media (V.E, 1975) de cada métrica resultado:

Red LAN y WAN 64KB como tamaño de los paquetes debido a que no hay restricciones en los tamaños:

$$T_{Paquete} = 64Kb$$

Ecuación 17 Tamaño el paquete de la red LAN y WAN entre Bogotá y Medellín

Red LAN, promedio de usuarios concurrentes:

$$U_{CLAN} = \frac{25 + 22 + 24 + 27 + 26}{5} = 24.8 \approx 24\ Usuarios$$

Ecuación 18 Cantidad promedio de usuarios concurrentes red LAN

Red WAN, promedio de usuarios concurrentes:

$$U_{CWAN} = \frac{88 + 90 + 75 + 87 + 89}{5} = 85.8 \approx 86\ Usuarios$$

Ecuación 19 Cantidad promedio de usuarios concurrentes red WAN

Buffer Red LAN y cantidad de paquetes:

$$B_{LAN} = \frac{\frac{16Mb}{64Kb}}{24_{UCLAN}} = \frac{\frac{16000Kb}{64Kb}}{24_{UCLAN}} = \frac{250}{24_{UCLAN}} = 10.41 \approx 10\ Paquetes\ por\ \ usuario$$

Ecuación 20 Calculo de la cantidad de paquetes que caben en el buffer por usuario red LAN

Buffer Red WAN y cantidad de paquetes:

$$B_{WAN} = \frac{\frac{500Mb}{64Kb}}{86_{UCWAN}} = \frac{\frac{500000Kb}{64Kb}}{86_{UCWAN}} = \frac{7812.5}{86_{UCWAN}} = 90.84 \approx 90\ Paquetes\ por\ \ enlace$$

Ecuación 21 Cálculo de la cantidad de paquetes que caben en el buffer para enlace WAN

Red LAN, promedio de tiempo de los ACK:

$$T_{ACK\ LAN}(ms) = \frac{2 + 2.2 + 1.3 + 1.1 + 0.8}{5} = 1.48 \approx 1.5ms$$

Ecuación 22 Cantidad promedio de tiempo de los ACK red LAN

Red WAN, promedio de tiempo de los ACK:

$$T_{ACK\ WAN}(ms) = \frac{5.1 + 4.8 + 4.6 + 5.4 + 5.3}{5} = 5.04 \approx 5ms$$

Ecuación 23 Cantidad promedio de tiempo de los ACK red WAN

Red LAN, promedio de ancho de banda por usuario:

$$BW_{LAN}(MB) = \frac{4.5 + 3.7 + 4.1 + 3.8 + 4.3}{5} = 4.08 \approx 4MB$$

Ecuación 24 Ancho de Banda promedio para cada usuario red LAN

Teniendo en cuenta que son 7 servicios, 6 TCP y 1 UDP el ancho de banda según lo que dicta el modelo se divide en 7 y se saca el servicio UDP asumiendo que los 6 servicios TCP de la red está en congestión mejor momento para aplicar el modelo:

$$BW_{LAN} = \frac{4(MB)}{7Servicios} = 0.57MB$$

$$BW_{6\ Servicios\ TCP} = 4MB - \left(\left({}^{4MB}/_{7Servicios}\right) * 5\right) = 1.142 \approx 1.5MB$$

Ecuación 25 Cálculo de ancho de banda de la red LAN con los 6 servicios TCP en congestión

Red WAN, promedio de ancho de banda por enlace:

$$BW_{WAN}(MB) = \frac{18.4 + 19.8 + 17.6 + 21.2 + 22.5}{5} = 19.97 \approx 20MB$$

Ecuación 26 Ancho de banda promedio para cada enlace red WAN

Teniendo en cuenta que son 7 servicios, 6 TCP y 1 UDP el ancho de banda según lo que dicta el modelo se divide en 7 y se saca el servicio UDP asumiendo que los 6 servicios TCP de la red está en congestión mejor momento para aplicar el modelo.

$$BW_{WAN} = \frac{80(MB)}{7Servicios} = 2.85MB$$

$$BW_{6\ Servicios\ TCP} = 20MB - \left(\left({}^{20MB}/_{7Servicios}\right) * 5\right) = 5.75MB$$

Ecuación 27 Cálculo de ancho de banda de la red WAN con los 6 servicios TCP en congestión

Después de realizar estos cálculos y a modo de resumen se obtienen:

	Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)
LAN	10	64	1.5	1.15
WAN	90	64	5	5.75

Tabla 20 Parámetros de la red real usados para la simulación de la misma

5.4 SIMULACIÓN DE LA RED SEGÚN LAS MEDICIONES OBTENIDAS

Como se mencionó con anterioridad la situación se realizó con ayuda del software ns2 realizando el siguiente modelo:

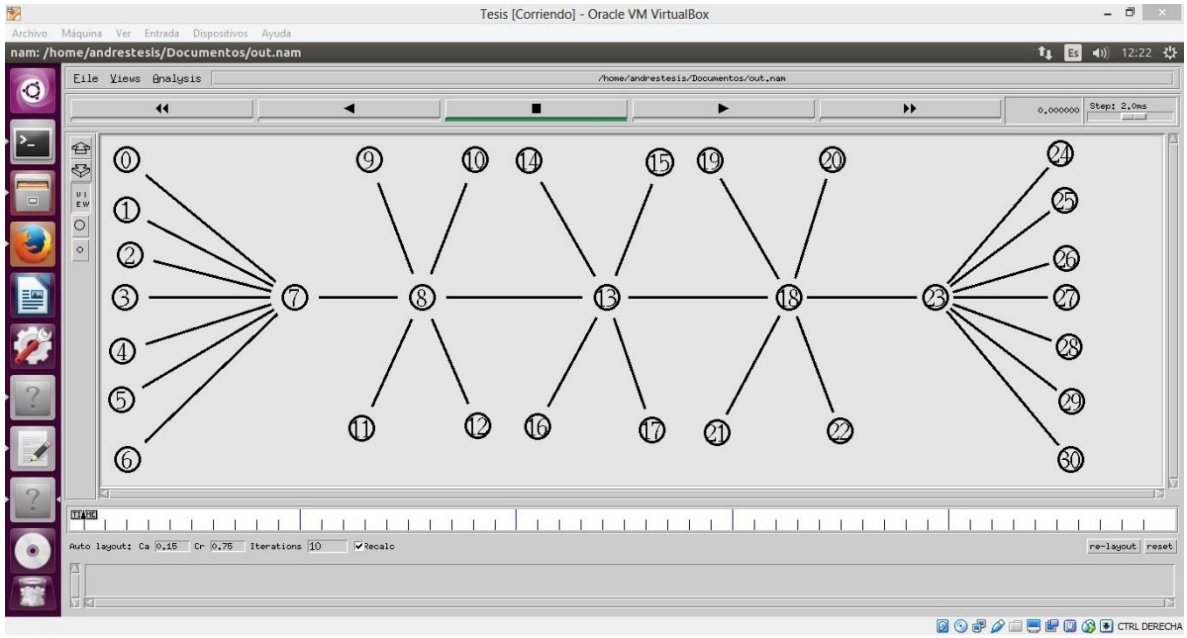


Figura 43 Modelo de red LAN-WAN-LAN a simular fuente: creación propia del esquema en ns2

Como se puede observar en uno de los extremos se encuentra una red LAN con los 7 servicios reportados que representaban el 95% del tráfico, el nodo 0 al 5 son los servicios TCP y el nodo 7 corresponde al servicio UDP, el mismo esquema se puede ver al otro extremo en donde el nodo 23 representa la otra red LAN que soporta los 7 servicios previamente mencionados, siendo del 24 al 29 los 6 servicios TCP y el nodo 30 el servicio UDP.

Se colocaron 3 nodos dentro de la red WAN, (8, 13 y 18) como se calculó anteriormente cada enlace tiene 80MB de ancho de banda y cada nodo usa 20MB para incluir esto en la simulación se implementaron 4 nodos extra, cada uno de 20MB a cada nodo WAN.

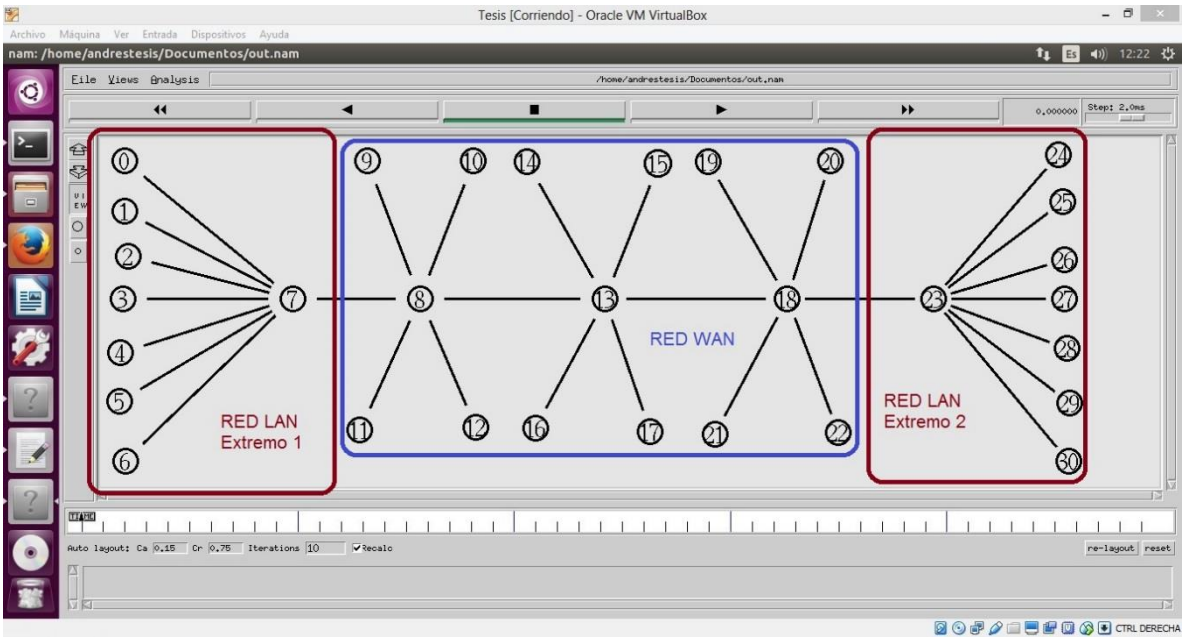


Figura 44 Componentes de la red LAN-WAN-LAN a simular fuente: creación propia del esquema en ns2

Una vez se realizó el modelo y este se implementó en los scripts para el simulador se ingresaron los datos obtenidos de la red real explicados en la sección anterior obtenido la siguiente simulación:

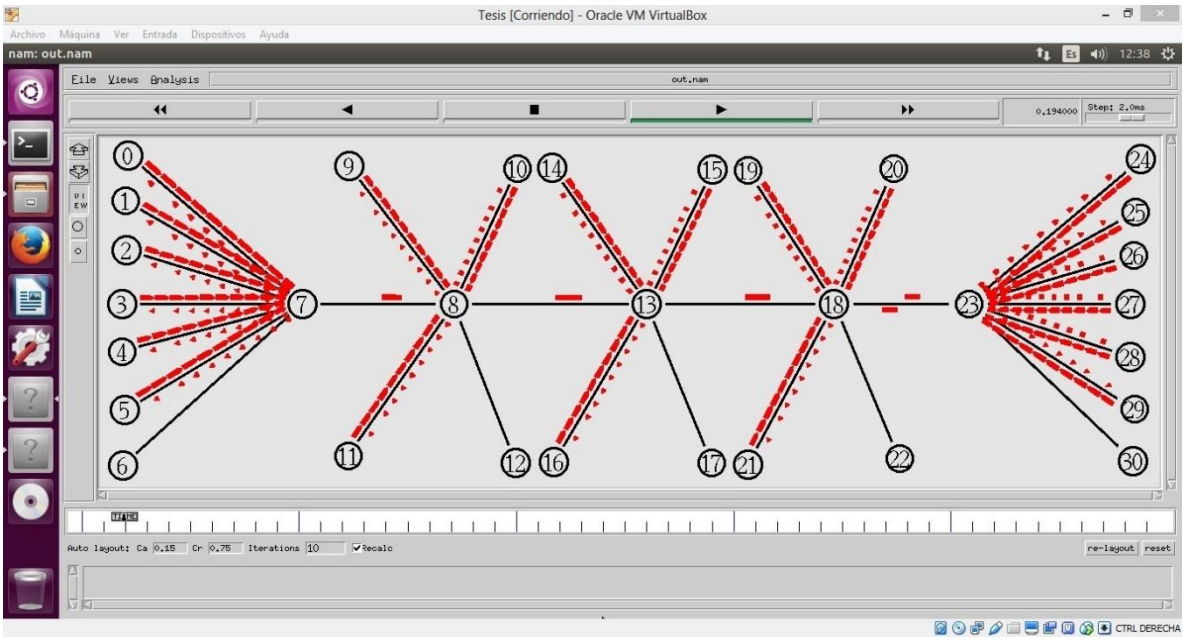


Figura 45 Simulación de la red real en ns2, fuente, creación propia del esquema en ns2

Al final de la simulación ns2 entrega un informe final de la simulación el cual indicó que la relación de error fue del 0.079%:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10	64	LAN 1.5	LAN 1.15	413762	330	0.079
WAN 90		WAN 5	WAN 5.75			

Tabla 21 Resultados obtenidos de la simulación de la red real, fuente creación propia

5.5 SIMULACIÓN DEL TOTAL DE ESCENARIOS

Como se mencionó anteriormente para este trabajo de investigación se obtuvieron un total de 36 escenarios según la cantidad de rangos y divisiones obtenidas, se realizó la simulación de la totalidad de ellos:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)
LAN 12 WAN 116	50	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75
LAN 11 WAN 101	40	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6
LAN 11 WAN 96	30	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45
LAN 10 WAN 88	20	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3
LAN 7 WAN 59	10	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15
LAN 6 WAN 58	1	LAN 0.45 WAN 1.45	LAN 4 WAN 20

Tabla 22 Rango de valores obtenidos para las métricas calculados siguiendo las instrucciones del modelo, fuente creación propia.

Escenario 1, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 12 WAN 116	50	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75	470029	270	0.057

Tabla 23 Valores de las métricas escenario 1

Escenario 2, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 12 WAN 116	50	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6	640703	348	0.054

Tabla 24 Valores de las métricas escenario 2

Escenario 3, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 12 WAN 116	50	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45	997092	426	0.042

Tabla 25 Valores de las métricas escenario 3

Escenario 4, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 12 WAN 116	50	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3	1185304	534	0.045

Tabla 26 Valores de las métricas escenario 4

Escenario 5, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 12 WAN 116	50	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15	1396546	690	0.049

Tabla 27 Valores de las métricas escenario 5

Escenario 6, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 12 WAN 116	50	LAN 0.45 WAN 1.45	LAN 4 WAN 20	1662292	912	0.054

Tabla 28 Valores de las métricas escenario 6

Escenario 7, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 101	40	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75	522722	330	0.063

Tabla 29 Valores de las métricas escenario 7

Escenario 8, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 101	40	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6	712556	390	0.054

Tabla 30 Valores de las métricas escenario 8

Escenario 9, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 101	40	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45	1132020	474	0.041

Tabla 31 Valores de las métricas escenario 9

Escenario 10, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 101	40	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3	1339150	588	0.043

Tabla 32 Valores de las métricas escenario 10

Escenario 11, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 101	40	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15	1599622	762	0.048

Tabla 33 Valores de las métricas escenario 11

Escenario 12, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 101	40	LAN 0.45 WAN 1.45	LAN 4 WAN 20	1856316	1014	0.054

Tabla 34 Valores de las métricas escenario 12

Escenario 13, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 96	30	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75	597559	336	0.05

Tabla 35 Valores de las métricas escenario 13

Escenario 14, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 96	30	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6	810503	402	0.04

Tabla 36 Valores de las métricas escenario 14

Escenario 15, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 96	30	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45	1055819	326	0.03

Tabla 37 Valores de las métricas escenario 15

Escenario 16, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 96	30	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3	1293831	369	0.028

Tabla 38 Valores de las métricas escenario 16

Escenario 17, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 96	30	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15	1815610	804	0.044

Tabla 39 Valores de las métricas escenario 17

Escenario 18, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 11 WAN 96	30	LAN 0.45 WAN 1.45	LAN 4 WAN 20	1828260	1134	0.062

Tabla 40 Valores de las métricas escenario 18

Escenario 19, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10 WAN 88	20	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75	711822	384	0.053

Tabla 41 Valores de las métricas escenario 19

Escenario 20, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10 WAN 88	20	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6	955156	456	0.047

Tabla 42 Valores de las métricas escenario 20

Escenario 21, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10 WAN 88	20	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45	1242902	284	0.022

Tabla 43 Valores de las métricas escenario 21

Escenario 22, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10 WAN 88	20	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3	1515302	320	0.021

Tabla 44 Valores de las métricas escenario 22

Escenario 23, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10 WAN 88	20	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15	1809070	417	0.023

Tabla 45 Valores de las métricas escenario 23

Escenario 24, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 10 WAN 88	20	LAN 0.45 WAN 1.45	LAN 4 WAN 20	2554607	1170	0.045

Tabla 46 Valores de las métricas escenario 24

Escenario 25, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 7 WAN 59	10	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75	945639	468	0.049

Tabla 47 Valores de las métricas escenario 25

Escenario 26, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 7 WAN 59	10	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6	1227079	552	0.044

Tabla 48 Valores de las métricas escenario 26

Escenario 27, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 7 WAN 59	10	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45	1584291	518	0.032

Tabla 49 Valores de las métricas escenario 27

Escenario 28, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 7 WAN 59	10	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3	1904329	685	0.035

Tabla 50 Valores de las métricas escenario 28

Escenario 29, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 7 WAN 59	10	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15	2157665	1038	0.048

Tabla 51 Valores de las métricas escenario 29

Escenario 30, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 7 WAN 59	10	LAN 0.45 WAN 1.45	LAN 4 WAN 20	2539769	1554	0.061

Tabla 52 Valores de las métricas escenario 30

Escenario 31, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 6 WAN 58	1	LAN 1.5 WAN 5	LAN 1.15 WAN 5.75	3758347	1508	0.04

Tabla 53 Valores de las métricas escenario 31

Escenario 32, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 6 WAN 58	1	LAN 1.29 WAN 4.29	LAN 1.72 WAN 8.6	4164893	2108	0.05

Tabla 54 Valores de las métricas escenario 32

Escenario 33, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 6 WAN 58	1	LAN 1.08 WAN 3.58	LAN 2.29 WAN 11.45	4535399	2808	0.061

Tabla 55 Valores de las métricas escenario 33

Escenario 34, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 6 WAN 58	1	LAN 0.87 WAN 2.87	LAN 2.86 WAN 14.3	4918497	3908	0.081

Tabla 56 Valores de las métricas escenario 34

Escenario 35, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 6 WAN 58	1	LAN 0.66 WAN 2.16	LAN 3.43 WAN 17.15	5630919	4874	0.086

Tabla 57 Valores de las métricas escenario 35

Escenario 36, corresponde a:

Buffer (Cantidad de paquetes por usuario)	Tamaño del paquete (KB)	Tiempo de ACK (ms)	Ancho de Banda (MB/s)	Total Paquetes Generados	Total Paquetes Perdidos	% de error
LAN 6 WAN 58	1	LAN 0.45 WAN 1.45	LAN 4 WAN 20	5845321	5174	0.089

Tabla 58 Valores de las métricas escenario 36

5.6 ANÁLISIS DE RESULTADOS

Tal como se esperaba en todos los casos en los que la cantidad del buffer disponible disminuía y por lo tanto el tamaño del paquete también lo hacía, la cantidad de paquetes aumentó, esto benefició a la red en el sentido de que hubo en proporción una menor pérdida de paquetes ya que un mayor número de estos lograban quedar en cola y no ser descartados.

Aun así, hubo paquetes que fueron descartados por no haber espacio en los buffers aunque en 33 de los 36 escenarios el número total de paquetes fue mayor, el porcentaje de error fue menor que los valores reales de la red de referencia.

Esto indica que el método de implementar un rango con valores fijos para los tamaños de los paquetes en función de la cantidad de buffer disponible, es más efectivo que él no controlar este aspecto y dejar que los paquetes tengan diferentes tamaños en función del servicio que prestan.

Para el caso de los paquetes que se dejaron con un tamaño fijo de 50KB con respecto a los valores reales de referencia se observó un aumento del número de estos en un 113.60% con una disminución de errores de trasmisión de 27.84%.

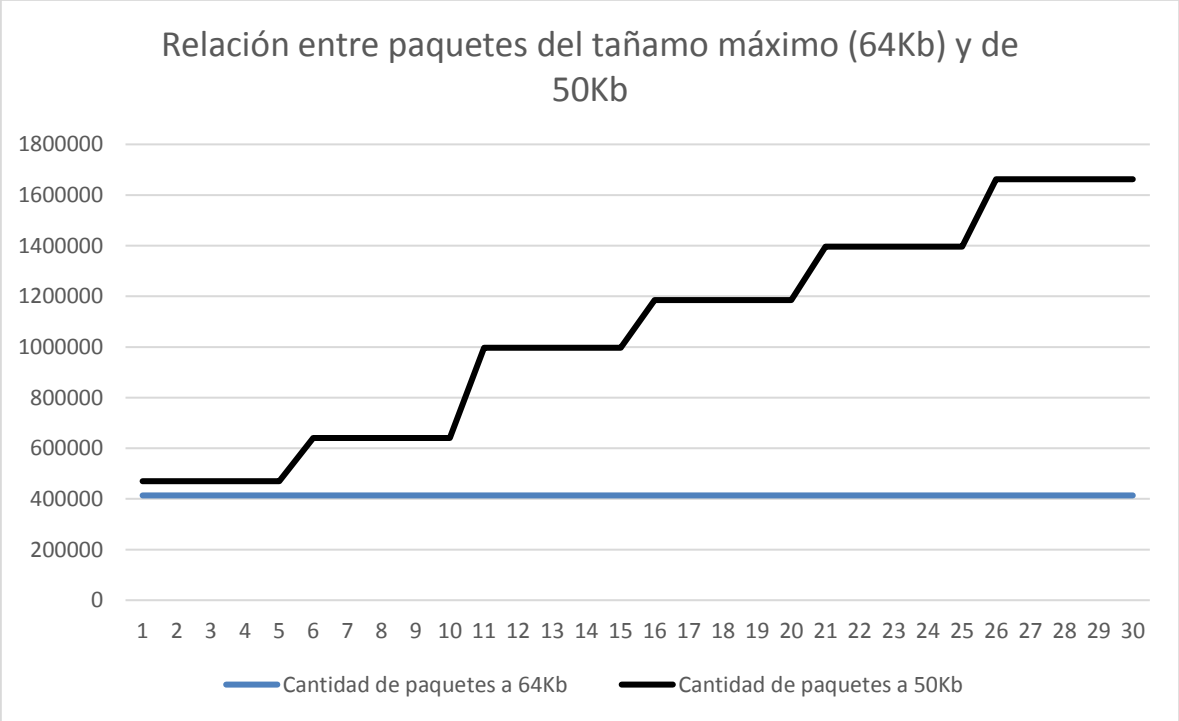


Figura 46 Relación entre paquetes del tamaño máximo (64KB) y de 50KB

Para el caso de los paquetes que se dejaron con un tamaño constante de 40KB con respecto a los valores reales de referencia se observó un aumento del número de estos en un 126.33% con una disminución de errores de trasmisión de 20.25%.

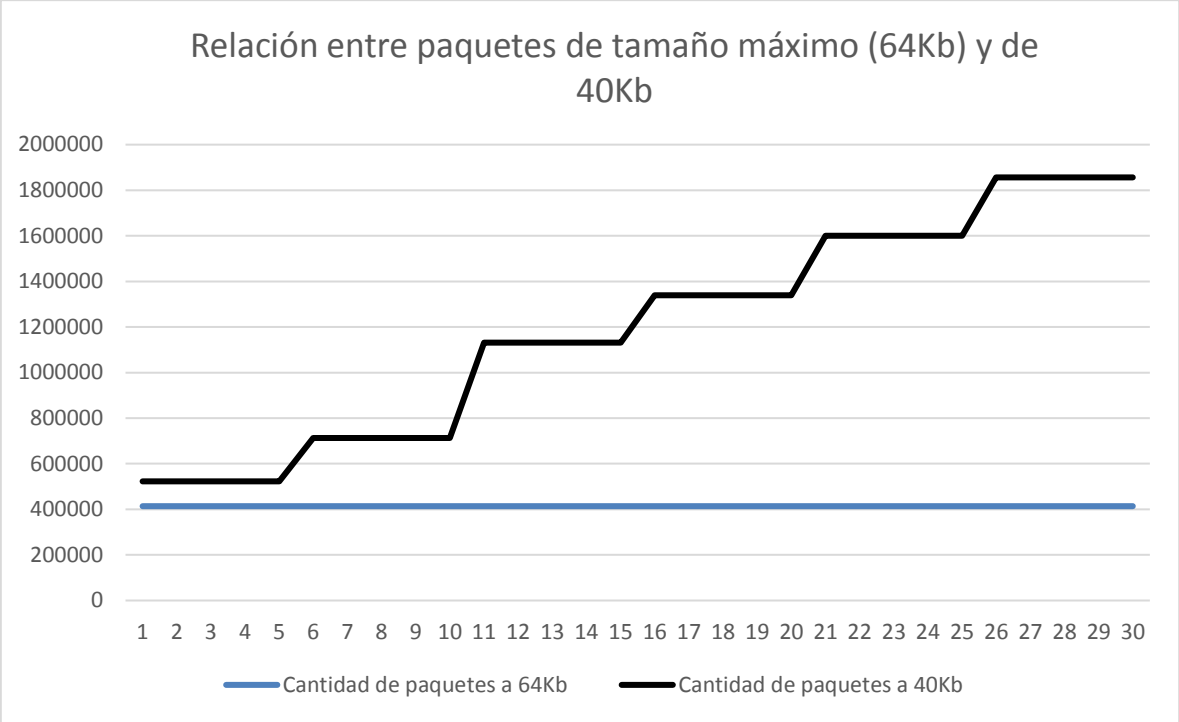


Figura 47 Relación entre paquetes del tamaño máximo (64KB) y de 40KB

Para el caso de los paquetes que se dejaron con un tamaño constante de 30KB con respecto a los valores reales de referencia se observó un aumento del número de estos en un 144.42% con una disminución de errores de trasmisión de 36.07%.

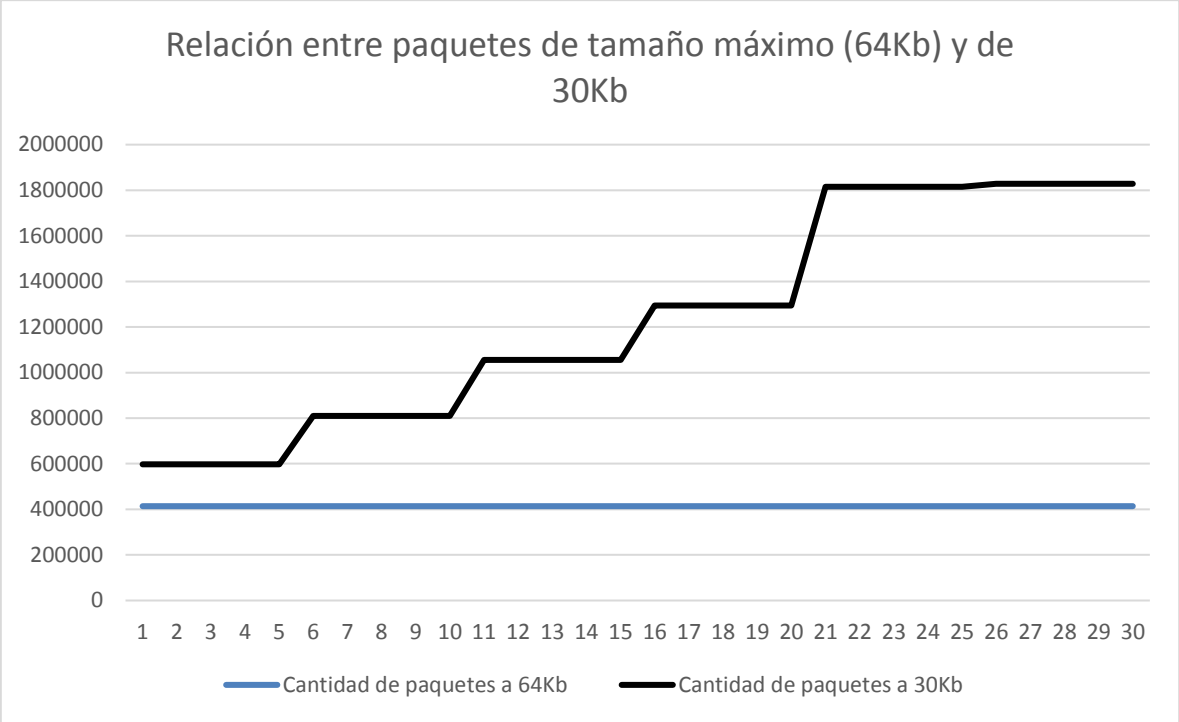


Figura 48 Relación entre paquetes del tamaño máximo (64KB) y de 30KB

Para el caso de los paquetes que se dejaron con un tamaño constante de 20KB con respecto a los valores reales de referencia se observó un aumento del número de estos en un 172.03% con una disminución de errores de trasmisión de 32.91%.

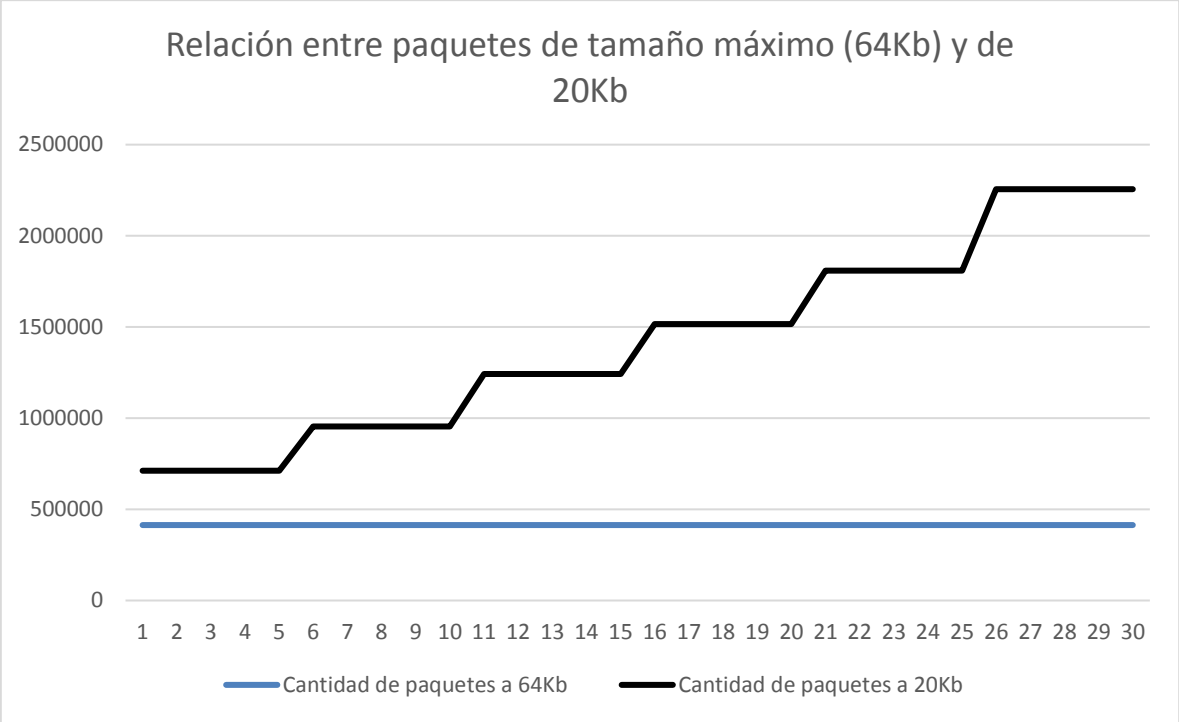


Figura 49 Relación entre paquetes del tamaño máximo (64KB) y de 20KB

Para el caso de los paquetes que se dejaron con un tamaño contante de 10KB con respecto a los valores reales de referencia se observó un aumento del número de estos en un 228.540% con una disminución de errores de trasmisión de 37.97%.

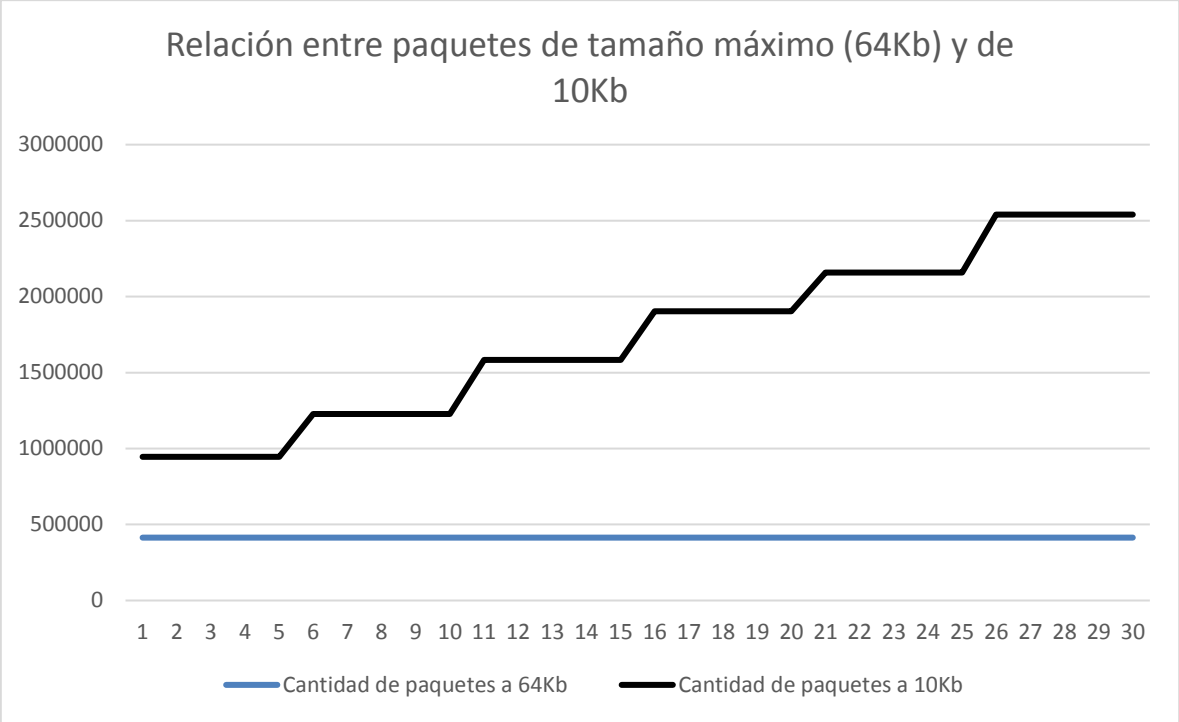


Figura 50 Relación entre paquetes del tamaño máximo (64KB) y de 10KB

Para el caso de los paquetes que se dejaron con un tamaño constante de 1KB con respecto a los valores reales de referencia se observó un aumento del número de estos en un 908.33% con una disminución de errores de trasmisión de 49.36%.

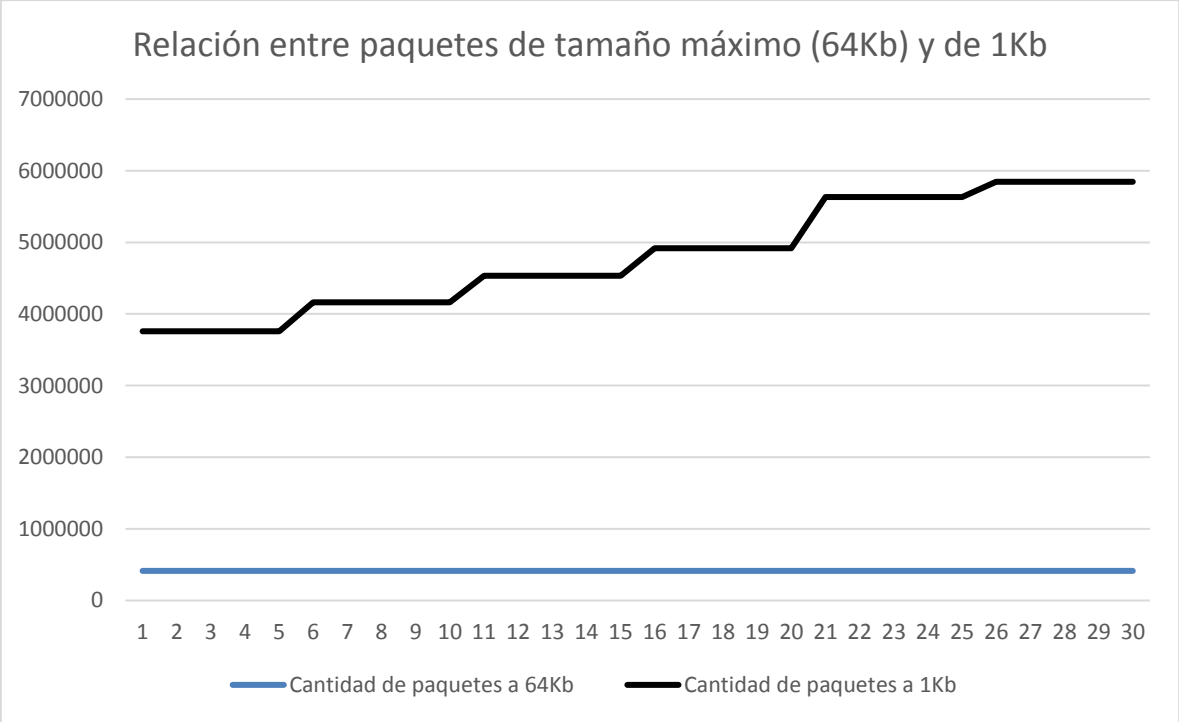


Figura 51 Relación entre paquetes del tamaño máximo (64KB) y de 1KB

También se observó que la metodología de variar el tamaño de los paquetes dentro de un rango definido es de igual manera más efectivo cuando el ancho de banda se mantiene constante, esto debido a que si este último no varía en el tiempo significa que la red presenta una limitante en el número de servicios que esta trasmitiendo, este factor puede llevar a aumentar el número de paquetes que podrían ser descartados y es por esto que variar el tamaño de estos genera una mayor estabilidad.

Para la totalidad de los escenarios con ancho de banda fijo los cuales corresponden a 6 en total, se observó que variando el tamaño del paquete el porcentaje de error de trasmisiones aunque era variante en todos los casos fue menor que los valores reales de referencia.

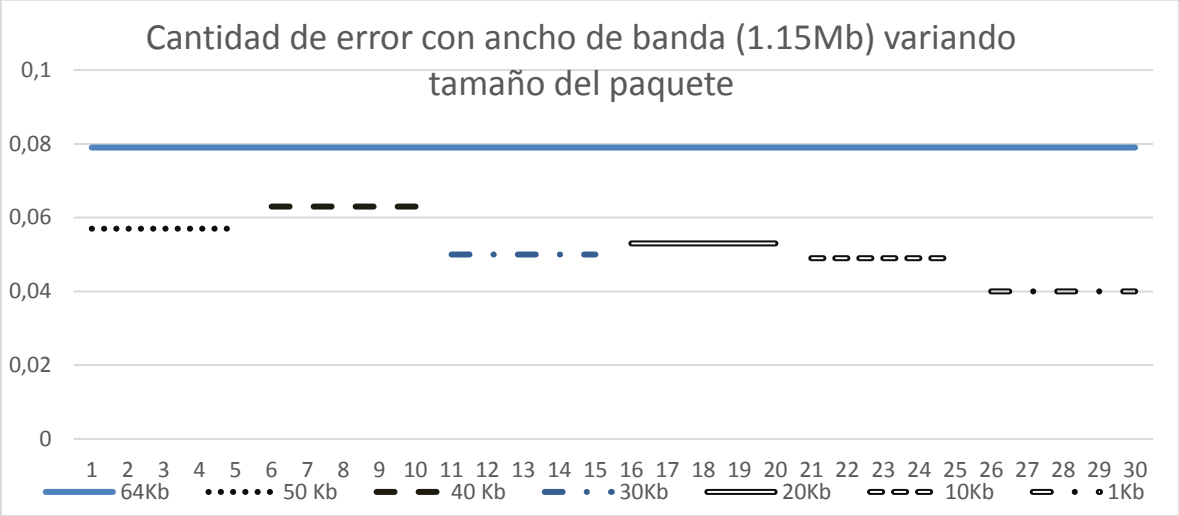


Figura 52 Cantidad de error con ancho de banda (1.15Mb) variando tamaño del paquete

Finalmente se observó que aplicando la combinación de métricas y los valores que el modelo le impone a estas, en 33 de los 36 escenarios el modelo fue más eficiente que la red real con la que se está comparando obteniendo las mejoras que se expresan a continuación:

Tabla 59 Comparación del porcentaje de mejora del modelo con respecto a la red real en cada uno de los escenarios

Red Real (Valores de control)	Porcentaje de errores de trasmisión de referencia 0.079	Porcentaje de mejora
Escenario 1	0.057	27.84
Escenario 2	0.054	31.64
Escenario 3	0.042	46.83
Escenario 4	0.045	43.03
Escenario 5	0.049	37.97

Escenario 6	0.054	31.64
Escenario 7	0.063	20.25
Escenario 8	0.054	31.64
Escenario 9	0.041	48.1
Escenario 10	0.043	45.56
Escenario 11	0.048	39.24
Escenario 12	0.054	31.64
Escenario 13	0.05	36.7
Escenario 14	0.04	49.36
Escenario 15	0.03	62.02
Escenario 16	0.028	64.55
Escenario 17	0.044	44.3
Escenario 18	0.062	21.51
Escenario 19	0.053	32.91
Escenario 20	0.047	40.5
Escenario 21	0.022	72.15
Escenario 22	0.021	73.41
Escenario 23	0.023	70.88
Escenario 24	0.045	43.03
Escenario 25	0.049	37.97
Escenario 26	0.044	44.3

Escenario 27	0.032	59.49
Escenario 28	0.035	55.69
Escenario 29	0.048	39.24
Escenario 30	0.061	22.78
Escenario 31	0.04	49.36
Escenario 32	0.05	36.7
Escenario 33	0.061	22.78
Escenario 34	0.081	Empeoro 2.5
Escenario 35	0.086	Empeoro 8.86
Escenario 36	0.089	Empeoro 12.65

Tabla 60 Comparación del porcentaje de mejora del modelo con respecto a la red real en cada uno de los escenarios

Tal como se mencionó solamente en 3 escenarios el modelo no fue más eficiente que la red real de hecho presentó un aumento del error de entre 2.5 y 12.65%, para estos 3 escenarios hay que tener en cuenta que son los 3 de mayor congestión, en donde el total de buffer disponible era el mínimo, el tamaño del paquete era de 1KB.

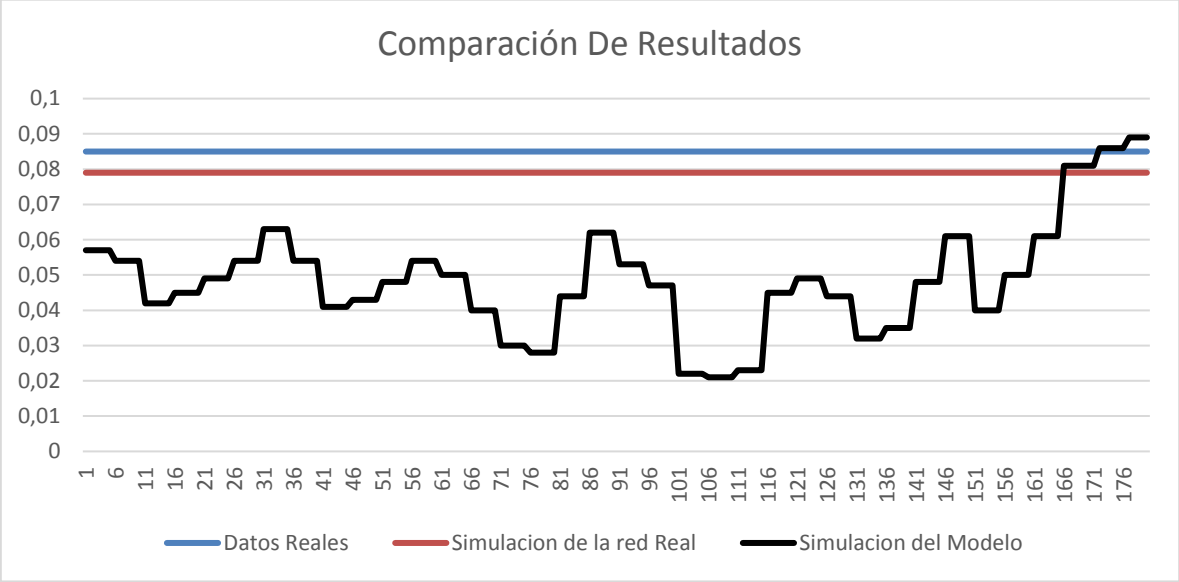


Figura 53 Comparación de la eficiencia del modelo con respecto a los datos reales y a la simulación de la red real

5.7 COMPARACIÓN CON TRABAJOS RELACIONADOS

Teniendo en cuenta los resultados previamente expuestos se procede a compararlos con trabajos anteriormente realizados con el fin de determinar la eficiencia total obtenida del modelo implementado en este trabajo de investigación:

El primer trabajo se llama Unleashing Tor, BitTorrent & Co How to Relieve TCP Deficiencies in Overlays de los autores Daniel Marks, Florian Tschorsch, Bjorn Scheuermann, en dicho trabajo los autores proponen como mecanismo para disminuir el control de la congestión el realizar links bidireccionales simétricos entre los extremos de la red así como realizar una priorización de los ACK usando herramientas de software sobre LINUX:

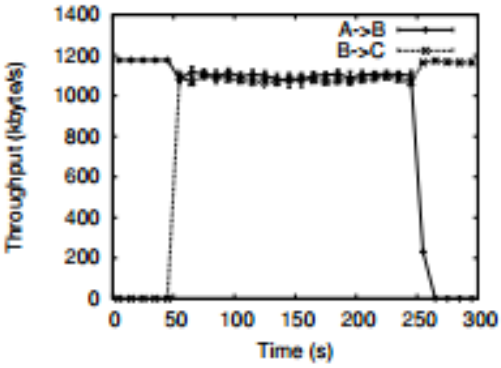


Fig. 2. Impact of ACK prioritization.

Figura 54 Resultados del control de congestión del paprer Unleashing Tor, BitTorrent & Co How to Relieve TCP Deficiencies in Overlays, fuente (Marks, Tschorsch, & Scheuermann, 2010)

Con este método los autores lograron un rendimiento del 90% en cuanto al rendimiento de la red.

El segundo trabajo se llama Agile-SD: A Linux-based TCP congestion control algorithm for supporting high-speed and short-distance networks de los autores Mohamed A. Alrshah, Mohamed Othman, Borhanuddin Ali, Zurina Mohd Hanapi, en este paper tocan el tema de congestión del protocolo TCP pero lo enfocan hacia las redes de distancias cortas (LAN) ya que según lo que se ha podido observar en los últimos años el número de estas redes han crecido considerablemente en todos los ámbitos posibles.

Aunque toman el tema de las redes LAN independientes también tienen en cuenta cuando estas redes tienen conexión directa o dependen de un centro de datos principalmente cuando estos últimos se encargan de realizar altos niveles de procesamientos de información o tienen altos niveles de tráfico a nivel de base de datos, los cuales también presentan niveles de congestión siendo un factor negativo ya que por su naturaleza se necesita un servicio eficiente.

Se menciona que principalmente estas redes presentan algoritmos de control de congestión que cuando alcanzan o se acercan considerablemente a un punto crítico asignan un ancho de banda que aunque es fijo es menor que el necesario lo cual deja de ser una solución viable en poco tiempo así como una disminución del buffer, para corregir este asunto los autores proponen implementar un algoritmo de control de congestión para redes de corta distancia este algoritmo se llama Agile-SD.

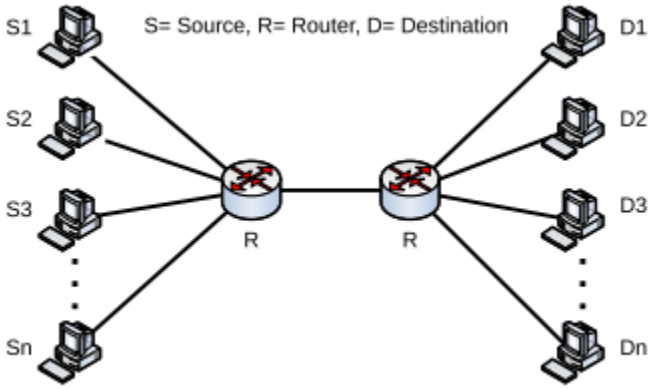


Fig. 8. Network topology with standard dumbbell bottleneck.

Figura 55 Topología de red sobre la que se probó el mecanismo del control de congestión, fuente: (Alrshah, Othman, Ali, & Hanapi, 2015)

Para ellos los autores realizaron diferentes tipos de simulaciones de redes LAN en donde compararon el algoritmo Agile-SD con algoritmos convencionales y con el Cubic ya que estos son los algoritmos más usados en los principales sistemas operativos.

Como resultado obtuvieron que el algoritmo Agile-SD presenta resultados un poco mejores que el promedio en condiciones normales pero resultados mucho mejores cuando se bajaba el buffer y ancho de banda, en esos casos fue cuando se presentó una pequeña tasa de error en trasmisión de paquetes, en conclusión, presentaron mejoras de promedio 40%.

Un tercer trabajo es llama DC-Vegas : A delay-based TCP congestion control algorithm for data center applications de los autores Jingyuan Wanga, Jiangtao Wenb, Chao Lia, Zhang Xionga, Yuxing Hand, en este se enfocan en el control de congestión para protocolo TCP

en las redes de centros de datos es muy diferente a los demás tipos de redes ya que los centros de datos requieren una buena latencia y estar en total capacidad de evitar colapsos cuando aumenta el rendimiento de manera considerable.

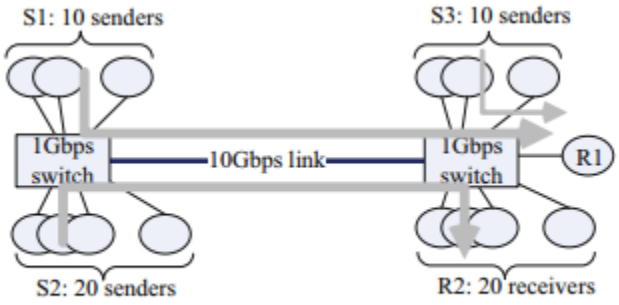


Fig. 17. Multi-bottleneck experiments topology.

Figura 56 Topología de red sobre la que se probó el mecanismo del control de congestión, fuente: (Wang, Wen, Li, Xiong, & Han , 2015)

Los autores haciendo un recordatorio de soluciones planteadas anteriormente mencionan al DCTCP, DCTCP2 y DCTCP3 además de todo tipo de inconvenientes a la hora de la implementación como el cambio de hardware toda la pila de protocolos que tienen los diferentes sistemas operativos cuando estos se cambian y otras aplicaciones de capa superior como balanceadores o firewalls.

Es por eso que los autores de este paper proponen un algoritmo de congestión para TCP llamado DC-Vegas el cual combina las ventajas de rendimiento del algoritmo DCTCP superando problemáticas de retardo convencionales.

Este algoritmo según los resultados obtenidos puede cumplir en tiempo real los requerimientos mínimos de reparto de datos en los centros de datos y requiere de una modificación mínima cada vez que se desee realizar un cambio significativo.

Esto lo probaron experimentando tanto con centros de datos reales como en simulaciones con Ns-2 en donde pudieron demostrar que tiene un rendimiento muy aproximado al que indicaba la teoría siendo un total de 80%.

Un cuarto trabajo se llama Exploring parallelism and desynchronization of TCP over high speed networks with tiny buffers de los autores Cheng Cui , Lin Xue , Chui-Hui Chiu , Praveenkumar Kondikoppa , Seung-Jong Park. en este paper los autores se basan en uno de los problemas que se presentan muy a menudo en redes tipo LAN-WAN-LAN que son las de mayor relevancia para el presente proyecto y es el tamaño del buffer de los diferentes routers por los que puede pasar un paquete, este problema ha sido tema de discusión durante ya largo tiempo en donde se discute cómo mejorar los enlaces en las redes de alta velocidad en el que el tamaño del buffer de los routers es ideal y por ser routers de alta velocidad se podría reducir el tamaño del buffer.

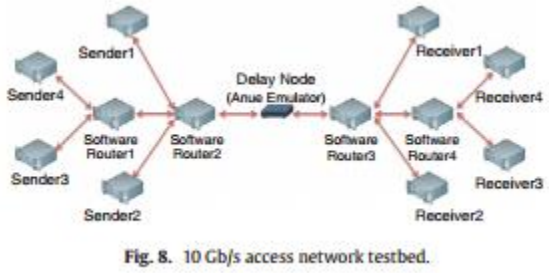


Figura 57 Topología de red sobre la que se probó el mecanismo del control de congestión, fuente: (Cui, Xue, Chiu, Kondikoppa, & Park, 2014)

Los principales problemas a atacar según los autores vendrían siendo 3: el exceso de ancho de banda, el tener ráfagas de datos en lugar de flujos y el que los paquetes sean asíncronos, aun así las investigaciones realizadas sugieren que estos 3 aspectos podrían no ser suficientes y que el tráfico TCP aun seguiría sufriendo retrasos y problemas con routers de buffers pequeños.

Para solucionar esto los autores proponen un algoritmo de control de congestión al que llamaron: “multicanal desincronizado TCP” el cual consiste en crear flujos con varios canales paralelos los cuales tienen total independencia entre sí en cuanto a sincronización adaptándose a las necesidades de los routers evitando problemas de tráfico principalmente en las ráfagas de datos.

En las simulaciones que ellos realizaron se encontró que aunque siguen existiendo cuellos de botella con el algoritmo de “multicanal desincronizado TCP” el cuello de botella en ráfagas de datos es de una magnitud 75% menor que los demás algoritmos de control de congestión en redes netamente ópticas.

Un último trabajo llamado TCP Vegas: End to End Congestion Avoidance on a Global Internet de los autores Lawrence S. Brakmo, Larry L. Peterson, en este paper los autores quieren mostrar el estado el arte del algoritmo de control de congestión TCP VEGAS, el cual tiene un mejor rendimiento que el TCP RENO, según datos de los autores este es mejor entre un 37% y un 71% además de solo presentar desde un quinto hasta un medio de pérdidas de paquetes.

Los autores muestran las 3 técnicas principales que el algoritmo TCP VEGAS implementa y presentan los resultados de sus experimentos utilizando una comparativa exhaustiva entre las simulaciones y mediciones reales en internet.

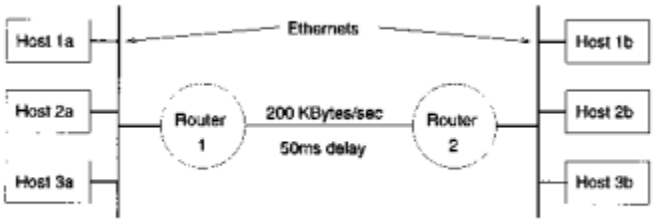


Fig. 6. Network configuration for simulations.

Figura 58 Topología de red sobre la que se probó el mecanismo del control de congestión, fuente: (Brakmo, O’Malley, & Peterson, 1994)

Dentro de sus principales hallazgos encontraron que se debe incluir un nuevo mecanismo de tiempo de espera, así como una nueva metodología que evite la congestión en donde se debe tratar de controlar el número de buffers adicionales que ocupan la conexión en la red y finalmente un mecanismo de comienzo lento modificado.

Trabajo	Mejora de la congestión
Unleashing Tor, BitTorrent & Co How to Relieve TCP Deficiencies in Overlays	90%
Agile-SD: A Linux-based TCP congestion control algorithm for supporting high-speed and short-distance networks	40%
DC-Vegas: A delay-based TCP congestion control algorithm for datacenter applications	80%
Exploring parallelism and desynchronization of TCP over high speed networks with tiny buffers	75%
TCP Vegas: End to End Congestion Avoidance on a Global Internet	37% - 71%
MODELO ALGORÍTMICO PARA LIMITAR LA CONGESTIÓN DEL PROTOCOLO TCP EN REDES DE EXTREMO A EXTREMO	***43% - 73%***

Tabla 61 Comparación de las mejoras del control de congestión del presente trabajo con otros realizados con anterioridad por otros autores, fuente: Creación propia

Como se puede ver en la tabla anterior el resultado del presente trabajo si bien no es el mejor comparado con otros autores, si está dentro del rango de porcentajes de mejoras que se presentaron, de igual manera el resultado mínimo tampoco fue peor, todas las mejoras de este trabajo están dentro del rango de los demás autores.

6. CONCLUSIONES, TRABAJOS FUTUROS Y APORTES DE LA INVESTIGACIÓN

6.1 CONCLUSIONES

Para lograr un control de la congestión en redes TCP de extremo a extremo es necesario tener en cuenta las principales métricas que determinan una buena calidad de servicio, más importante aún es la relación que se genera entre el Buffer y los tamaños de los paquetes al igual que entre los tiempos de respuestas y el ancho de banda disponible, el cómo se monitorea y se asignan valores en estas relaciones es lo que permite dicho control.

Es necesario enfatizar que estas métricas por si solas pueden indicar diversos niveles de calidad de servicio, por ejemplo, tiempos cortos de ACK no necesariamente indica que se están usando todos los recursos de la red, o poco espacio en los buffers de por sí solo indica una posible congestión, pero si éstas se relacionan entre si se lograrán resultados más adecuados, ya que este tipo de red tiene comportamientos diferentes a otros tipos y tecnologías de conexión.

Es necesario una vez se logran relacionar las métricas entre si el llevar un constante monitoreo de éstas ya que el comportamiento de las redes puede ser muy variante según factores externos a menudo no controlables, dicho monitoreo logrará que se realicen los cambios necesarios en el momento necesario.

Tal como se mencionó para lograr un correcto funcionamiento del modelo es necesario relacionar las métricas de éste, pero se debe ser cuidadoso al momento de hacerlo, ya que hay métricas que no se pueden relacionar con otras, o también se puede dar el caso de métricas que dan mejores resultados cuando se relacionan con otras específicas, más exactamente se realizó la relación proporcional entre la cantidad de espacio libre en los buffers y el tamaño de los paquetes, y una relación inversamente proporcional entre los tiempos de los ACK y el ancho de banda habilitado.

La totalidad de buffer disponible es la métrica que determinará en proporción el tamaño del paquete, esto debido a que entre mayor buffer disponible paquetes de tamaño más grande se podrán recibir sin riesgo de ser rechazados, esto hace que las trasmisiones se completen más rápidamente, lo que despeja el canal con mayor prontitud. El comportamiento de estos rangos se distribuyeron dejando el ideal de entre un 100% y un 1% libre ya que por un lado un buffer con 100% de tamaño libre es tan poco común que se debe sacar de los parámetros, y un buffer de 1% libre es una congestión extrema en la cual ni el modelo de este trabajo de investigación ni los algoritmos de control de congestión existentes pueden solventar totalmente, es por esto que el modelo toma de entre 90%y 70% como una red que opera sin contratiempos, entre un 50% y un 30% una red que opera establemente y entre un 10% y un 1% una red que presenta congestión.

El poner fijo de entre un rango determinado el tamaño que tendrán los paquetes es un método para controlar la congestión con mayor facilidad ya que no se depende de la aleatoriedad que cada tipo de servicio puede presentar, esta hace que los sistemas sean impredecibles y en varias ocasiones extremadamente difíciles de controlar para el modelo se dejó de un lado el tamaño máximo permitido de 64KB debido a que este tamaño no se alcanza tan a menudo, además de que el modelo justamente para su buen funcionamiento indica que debe ser menor que el máximo, siendo el nuevo tope máximo de 50KB hasta un mínimo de 1KB.

Debido a que al ACK y su correspondiente tiempo es un factor determinante y característico del protocolo TCP se debe entender que la relación entre la cantidad de tiempo es en proporción la que determina el ancho de banda, ya que el que se presenten respuestas rápidas indica que el medio de trasmisión soportará una mayor cantidad de envíos de diferentes naturalezas y por lo tanto se puede aumentar el ancho de banda, para esto se toman valores promedio de referencia de la red en la que se quiere aplicar el modelo obteniendo para el caso puntual de aplicación de este trabajo de investigación un total de 1.5ms para el caso de la red LAN y 5ms para el caso de la red WAN.

Aunque el modelo está diseñado con un enfoque de redes TCP de extremo a extremo no se debe obviar otro tipo de protocolos de alto impacto que al igual que TCP se encuentran presentes en capa 4 como lo es el protocolo UDP, estos protocolos extra de capa 4 que no son TCP se tuvieron en cuenta durante el diseño del modelo y posteriores simulaciones ya que en la realidad son factores que generan aún más tráfico y posibles congestiones de entre todos los puertos de los que puede hacer uso UDP se tomaron en cuenta el 69 y el 161 ya que eran los usados en red real sobre la cual se simuló el modelo.

El método presenta su desempeño más alto cuando se logra encontrar un punto de equilibrio entre la relación de todas las métricas contempladas, dicho punto de equilibrio debe tener en cuenta que el tamaño de los paquetes en la cola del buffer no debe ser muy pequeña ya que esto generará que la cola sea muy larga y cada paquete deba esperar más tiempo del debido para ser procesado, pero tampoco debe ser muy grande ya que genera poco espacio de almacenamiento en buffer lo que puede ocasionar pérdida de paquetes, este “punto de equilibrio” se encontró entre el rango de tamaños de paquetes de entre 20KB y 50KB en todos los casos con un ancho de banda para la red LAN de entre 2.29MB/s y 2.86MB/s, estos valores de ancho de banda corresponden a darle prioridad a la mitad de los servicios más concurrentes que la red presentaba, el tamaño del paquete hizo que el porcentaje de error variara de entre 43.03% y 73.41%, valores que fueron los mejores con respecto a los demás escenarios.

Uno de los aspectos más difíciles de controlar y que trata de solventar el modelo lo mejor posible es el hecho de que en la red WAN la cantidad de nodos es desconocida, así como sus características técnicas lo cual es uno de los principales factores a tener en cuenta al momento de la implantación y configuración del modelo, si se desconoce este dato, se debe realizar un monitoreo aún más exhaustivo de las métricas con el fin de determinar el mejor comportamiento de la red, para este caso después de 5 días de seguimientos se determinó que la red WAN presentaban nodos con equipos Cisco con 500MB de buffer, 80MB de ancho de banda dividido en 4 canales cada uno de 20MB y un promedio de tiempo de ACK de 5ms.

El protocolo TCP, aunque ya lleva una considerable cantidad de años es aun usado debido a su confiabilidad, aspecto que por ejemplo UDP no ofrece al mismo nivel, este se puede visualizar en el hecho de que en conexiones tan grandes como las de extremo a extremo que literalmente pueden ser a escala intercontinental aún se use con todas sus limitantes.

Es claro que ésta es apenas la primera versión de un modelo como este, debido a que TCP en su propia arquitectura al momento de funcionar por su propia naturaleza generará congestión su mecanismo de seguridad y confiabilidad a la hora de la entrega de la información hacen que la congestión sea algo que esté siempre presente.

Otro factor que hace necesaria la generación de este tipo de modelos aparte de la generación natural de congestión de TCP es la evolución de la internet, a medida que pasan los años surgen nuevos servicios, nuevas trasmisiones y nuevas integraciones con otras tecnologías las cuales aumentan el número de hosts esto hace que constantemente se vea la necesidad de creaciones de nuevos y más efectivos modelos.

Aunque según las evidencias que se pueden ver en las simulaciones es claro que el modelo podría funcionar mucho mejor si se llevan a cabo buenas prácticas de redes y monitoreo de las mismas tales como mejorar aspectos como subneting, jitter, firewalls, proxy las cuales en conjunto con el modelo lograrán limitar la congestión a niveles sumamente bajos lo que aumentaría la calidad del servicio.

6.2 TRABAJOS FUTUROS

1. Mejoras del modelo para que se pueda usar con absolutamente todas las versiones de TCP
2. Mejoras del modelo para que trabaje en conjunto con algoritmos vigentes de control de congestión
3. Diseño de modelos que funcionen bajo redes neuronales de tal manera que puedan evolucionar constantemente y de manera autónoma

4. Basados en las experiencias de diseñar varios modelos llegar a la capacidad de diseñar un algoritmo de control de congestión que sea parte del protocolo TCP a nivel de programación

6.3 APORTES DE LA INVESTIGACIÓN

- Se realizó una investigación del estado del arte sobre los actuales factores que generan congestión en redes de extremo a extremo
- Se realizó una investigación acerca de los factores presentes en la internet actual que generan congestión, así como una serie de recomendaciones para mejorar la red y que además sirvieron como base para la elaboración del modelo
- Un modelo algorítmico que reduce la congestión del protocolo TCP evitando retrasos, retrasmisiones, trasmisiones fallidas y errores en aplicaciones.
- Un protocolo de pruebas que permita una correcta implementación, puesta en operación y recolección de resultados para las simulaciones de este modelo algorítmico que garanticen la calidad de los mismos

7. REFERENCIAS

Abed, G. A., Ismail, M., & Jumari, K. (2012). Exploration and evaluation of traditional TCP congestion control techniques. *Journal of King Saud University – Computer and Information Sciences*.

Alrshah, M., Othman, M., Ali, B., & Hanapi, Z. M. (2015). Agile-SD: A Linux-based TCP congestion control algorithm for supporting high-speed and short-distance networks. *Journal of Network and Computer Applications*.

Avrachenkova, K., Ayestab, U., Doncelc, J., & Jacko, P. (2012). Congestion control of TCP flows in Internet routers by means of index policy. *Journal of Network and Computer Applications*.

Ibm, R.U., Rosu, D.: An Evaluation of TCP Splice Benefits in Web Proxy Servers. In: WWW, ACM Press (2002)

Azana Hafizah Mohd Aman, A.-H. A. (2016). Network Simulators Parametric Comprasion for Network Mobility Management. *International Journal Of Future Generation Communication and Networking*, 17-28.

Bing, B. (2001). *Wireless LANs and Home Networks.* Singapore: World Scientific Publishing Co. Pte. Ltd.

Bing, B. (2010). *3D and HD Broadband Video Networking.* Norwood,: ARTECH HOUSE.

Maltz, D.A., Bhagwat, P.: TCP Splicing for Application Layer Proxy Performance. Technical report, IBM Research Report 21139 (Computer Science/Mathematics) (1998)

Bogdan Moraru, F. C. (2001). *Practical Analysis of TCP Implementations: Tahoe, Reno, NewReno.*

Brakmo, L. S., O'Malley, S., & Peterson, L. L. (1994). *TCP Vegas: New Techniques for Congestion Detection and Avoidance.* Arizona: Department of Computer Science The University of Arizona.

Briceño, J. E. (2005). *TRANSMISIÓN DE DATOS.* Merida: Taller de Publicaciones de la Facultad de Ingeniería, ULA.

Akamai: Akamai's EdgePlatform for Application Acceleration. Akamai, Inc., 2007.

Cisco. (01 de 04 de 2017). *Cisco.com LAN*. Obtenido de http://www.cisco.com/c/en/us/products/collateral/routers/network-convergence-system-5000-series/datasheet-c78-736153.pdf

Cisco. (01 de 04 de 2017). *Cisco.com WAN*. Obtenido de http://www.cisco.com/c/en/us/products/collateral/routers/900-series-industrial-routers/datasheet-c78-732129.html

Tariq, M., Zeitoun, A., Valancius, V., Feamster, N., Ammar, M.: Answering What-If Deployment and Configuration Questions with WISE. In: ACM SIGCOMM.

Clark, D. D., & Kleinrock , L. (2009). A Brief History of the Internet. *ACM SIGCOMM Computer Communication Review*, 22 - 31.

Cohen-Almagor,, R. (2011). Internet History. *International Journal of Technoethics*, 45-64.

Corner, D. E. (2000). *Redes Globales de Informacion con Internet y TCP/IP.* Prentice Hall.

Huang, C., Wang, Y.A., Li, J., Ross, K.W.: Measuring and Evaluating Large-Scale CDNs. In: MSR Technical Report MSR-TR-2008-106.

Cui, C., Xue, L., Chiu, C.-H., Kondikoppa, P., & Park, S.-J. (2014). Exploring parallelism and desynchronization of TCP over high speed networks with tiny buffers. *Journal of Network and Computer Applications*.

Disanzo, M., & Simón, J. L. (2006). Redes Peer to Peer y Tecnología JXTA. *Monografia*. Cátedra de Sistemas Distribuidos Escuela de Ingeniería Electrónica Departamento de Sistemas e Informática .

Ekiz, N., Rahman, A. H., & Amer, P. (2011). *Misbehaviors in TCP SACK Generation.* Newark, Delaware: ACM SIGCOMM Computer Communication Review.

Wang, Y.A., Huang, C., Li, J., Ross, K.W.: Measuring Network Performance for Cloud Services with AdMeasure. (2009) Submitted.

Elias Weingartner, H. v. (2009). A performance comparison of recent network simulators. *IEEE ICC* .

Feit, S. (2004). *TCP/IP: Arquitectura, Protocolos e Implementacion con IPV6 Y Seguridad de IP.* S.A. MCGraw-Hill.

Garcia Sanjuan, L. M. (2009). *Diseño e Implementacion de un protocolo de redes PEER-TO-PEER.* Madrid: Universidad Pontificia Comillas.

Allman, M., Floyd, S., Partridge, C.: Increasing TCP's Initial Window. RFC 3390

Giambene, G. (2005). *QUEUING THEORY AND TELECOMMUNICATIONS Networks and Applications.* Roma: Springer.

Hengartner, J., Bolliger, J., & Gross, T. (2000). *TCP Vegas Revisited.* Pittsburgh: School of Computer Science Carnegie Mellon University Pittsburgh.

Hengartner, J., Bolliger, J., & Gross, T. (s.f.). *TCP Vegas Revisited.* Pittsburgh: School of Computer Science Carnegie Mellon University Pittsburgh.

Krishnan, R., Madhyastha, H.V., Srinivasan, S., Jain, S., Krishnamurthy, A., Anderson, T., Gao, J.: Moving Beyond End-to-End Path Information to Optimize CDN Performance. In: ACM IMC.

Informatica, C. C. ((Consultado 16 de Noviembre de 2015) de Junio de 2014). *CCM Comunidad Informatica*. Obtenido de http://es.ccm.net/contents/281-protocolo-tcp

isi.edu. (02 de 03 de 2017). *NSNAM*. Obtenido de http://nsnam.sourceforge.net/wiki/index.php/User_Information#The_Network_Simulator_-_ns-2

jist. (04 de 04 de 2017). *jist.ece.cornell.edu*. Obtenido de http://jist.ece.cornell.edu/index.html

Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. RFC 3390.

Jitendra Padhye, V. F. (2000). *Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation.* IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 8, NO. 2.

Keefer, A., & Baiget, T. (2009). How it all began: a brief history of the Internet. *VINE*, 90 - 95.

Keefer, A., & Baiget, T. (s.f.). How it all began: a brief history of the Internet. *VINE*, 90 - 95.

Allman, M.: Tcp byte counting refinements. SIGCOMM Comput. Commun

Kuchler, A., & Schapranow, M. P. (2005). Congestion Control. *Seminar Communication Networks*, 1-31.

Kurose, J. F., & Ross , K. W. (2004). *Redes de Computadores un enfoque descedente basado en internet* (Segunda ed.). Pearson.

Lawrence , B. S., & Peterson, L. L. (1995). TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 13*.

Huang, Y., Mehrotra, S., Li, J.: A Hybrid FEC-ARQ Protocol for Low-Delay Lossless Sequential Data Streaming. In: ICME.

Luoa, H., Ci, S., Wu, D., & Tang, H. (2010). End-to-end optimized TCP-friendly rate control for real-time video streaming over wireless multi-hop networks. *Journal of Visual Communication and Image Representation*.

Luque Calderón, A. F., & Vela Porras, E. J. (2015). SEGURIDAD, RENDIMIENTO Y QOS EN TCP. *Universidad Distrital Francisco José de Caldas*, 1-38.

Luque Calderon, A. F., & Vela Porras, E. J. (2016). Prediccion de trafico a traves de redes neuronales artificiales. *Universidad Distrital Francisco Jos´e de Caldas*, 1-10.

Padmanabhan, V.N., Mogul, J.C.: Improving HTTP Latency. In: WWW Conference.

Marks, D., Tschorsch, F., & Scheuermann, B. (2010). Unleashing Tor, BitTorrent & Co.: How to Relieve TCP Deficiencies in Overlays. *35th Annual IEEE Conference on Local Computer Networks*, 320-323.

Navarro Gavira, S. (2001). *e-reding*. Obtenido de bibing.us.e: http://bibing.us.es/proyectos/abreproy/11306/fichero/TEORIA%252F09+-+Capitulo+4.pdf

nsman. (04 de 04 de 2017). *www.nsnam.org*. Obtenido de https://www.nsnam.org/overview/what-is-ns-3/

Obata, H., Nishimoto, S., & Ishida, K. (2009). TCP Congestion Control Method of Improving Friendliness over Satellite Internet, information, Communications and Signal Processing. *ICICS 2009. 7th International Conference on, School of Information Sciences Hiroshima, Japan*.

omnetpp. (04 de 04 de 2017). *omnetpp.org*. Obtenido de https://omnetpp.org/

Practicas de Redes, Protocolo TCP. (2003). Extremadura, España: Universidad de Extremadura.

Mogul, J.C.: The Case for Persistent-Connection HTTP. ACM CCR

Rath, H. K., & Karandikar, A. (2011). Performance Analysis of TCP and UDP-based Applications in a IEEE 802.16 deployed Network. *Wireless Personal Multimedia Communications (WPMC), 2011 14th International Symposium on, 2011, TCS Networks Lab, Bangalore 560 066, India*.

Rhee, I., & Xu, L. (2008). *CUBIC: A New TCP-Friendly High-Speed TCP Variant.* New York: ACM SIGOPS Operating Systems Review.

Salazar, I. A. (2007). *Métodos Para El Mejoramiento Del Desempeño Del Protocolo Tcp/Ip Sobre Redes Inalámbricas.* Guatemala: Universidad de San Carlos de Guatemala.

Santín González, A. (2007). *Peer 2 Peer Sistemas Operativos Distribuidos.* http://www.dit.upm.es/.

Cohen, E., Krishnamurthy, B., Rexford, J.: Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters. ACM CCR

scalable-networks. (04 de 04 de 2017). *http://web.scalable-networks.com*. Obtenido de http://web.scalable-networks.com/simulation-platform-architecture

Shivangi Surati, D. C. (2016). A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. *Engineering Science and Technology, an International Journal*.

Spiegel, M. (1975). *Teoria y Problemas de Probabilidad y Estadistica.* New York: Mc Graw Hill.

Stallings, W. (2004). *Comunicaciones y redes de computadores.* Pearson Educación.

Tanenbaum, A. (2003). *Redes De Computadoras.* Amsterdam: Prentice Hall 4º Edicion.

Utsumi, S., Muhammad , S., & Zabir, S. (2014). A new high-performance TCP friendly congestion control over wireless networks. *Journal of Network and Computer Applications*.

V.E, G. (1975). *TEORIA DE LAS PROBABILIDADES Y ESTADISTICA MATEMATICA .* Madrid: MIR.

Wang, J., Wen, J., Li, C., Xiong, Z., & Han , Y. (2015). DC-Vegas: A delay-based TCP congestion control algorithm for datacenter applications. *Journal of Network and Computer Applications*, 103-114.

Usability Engineering, Jakob Nielsen (1993)

Card, S. K., Robertson, G. G., and Mackinlay, J. D. (1991). The information visualizer: An information workspace.*Proc. ACM CHI'91 Conf.* (New Orleans, LA, 28 April-2 May), 181-188.

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proc. AFIPS Fall Joint Computer Conference* **Vol. 33**, 267-277.

Myers, B. A. (1985). The importance of percent-done progress indicators for computer-human interfaces. *Proc. ACM CHI'85 Conf.* (San Francisco, CA, 14-18 April), 11-17.

ANEXOS

Script con la programacion del modelo en lenguaje TCL para el simuador ns2:

```
#Creacion del objeto simulador
set ns [new Simulator]

#Definicion de distintos colores para la simulacion, es opcional, pero recomendable, siempre con el objeto $ns, siempre se empieza en cero
$ns color n0 Blue
$ns color n1 Red
$ns color n2 Yellow
$ns color n3 Green

#Abrir un archivo para escritura de formato out.nam. Esto para enviar el trazado de la simulacion. Se crea como objeto $nf.
set nf [open out.nam w]

#Aca se envia todo el trazado al archivo
$ns namtrace-all $nf

#El trazado anterior es poco entendible ya que es mas cercano al lengaje maquina, pero debe ser creado para que el programa nam lo interprete. Sin embargo, hay otro tipo de trazado que es mas legible para nosotros. Para ello es necesario añnadir las siguientes lineas
set tf [open out.tr w]
$ns trace-all $tf

#Definicion del procedimiento 'finish', es el que se llama cuando finaliza la simulacion.
proc finish {} {
        global ns nf
        # Refrescar el trazado
        $ns flush-trace
        # Cerrar el objeto $nf de trazado
        close $nf
        # Ejecutar .nam resultante
        exec nam out.nam &
        exit 0
}

# Desde aca se inserta el codigo personalizado
```

```
#Crear los 31 nodos como n0, n1, n2, n3..., n31 siempre se empieza en cero
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
set n10 [$ns node]
set n11 [$ns node]
set n12 [$ns node]
set n13 [$ns node]
set n14 [$ns node]
set n15 [$ns node]
set n16 [$ns node]
set n17 [$ns node]
set n18 [$ns node]
set n19 [$ns node]
set n20 [$ns node]
set n21 [$ns node]
set n22 [$ns node]
set n23 [$ns node]
set n24 [$ns node]
set n25 [$ns node]
set n26 [$ns node]
set n27 [$ns node]
set n28 [$ns node]
set n29 [$ns node]
set n30 [$ns node]

#Definicion de los links entre cada nodo Por ejemplo, el primero dice que en el objeto $ns, los nodos $n0 y $n1 tendran un link bidireccional de 1.15Mbps, con un retardo de 1.1ms y el tipo de cola sera DropTail
$ns duplex-link $n0 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n1 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n2 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n3 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n4 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n5 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n6 $n7 1.15Mb 1.5ms DropTail
$ns duplex-link $n7 $n8 5.75Mb 5ms DropTail
```

```
$ns duplex-link $n8 $n9 1.15Mb 1.5ms DropTail
$ns duplex-link $n8 $n10 1.15Mb 1.5ms DropTail
$ns duplex-link $n8 $n11 1.15Mb 1.5ms DropTail
$ns duplex-link $n8 $n12 1.15Mb 1.5ms DropTail
$ns duplex-link $n8 $n13 5.75Mb 5ms DropTail
$ns duplex-link $n13 $n14 1.15Mb 1.5ms DropTail
$ns duplex-link $n13 $n15 1.15Mb 1.5ms DropTail
$ns duplex-link $n13 $n16 1.15Mb 1.5ms DropTail
$ns duplex-link $n13 $n17 1.15Mb 1.5ms DropTail
$ns duplex-link $n13 $n18 5.75Mb 5ms DropTail
$ns duplex-link $n18 $n19 1.15Mb 1.5ms DropTail
$ns duplex-link $n18 $n20 1.15Mb 1.5ms DropTail
$ns duplex-link $n18 $n21 1.15Mb 1.5ms DropTail
$ns duplex-link $n18 $n22 1.15Mb 1.5ms DropTail
$ns duplex-link $n18 $n23 5.75Mb 5ms DropTail
$ns duplex-link $n23 $n24 1.15Mb 1.5ms DropTail
$ns duplex-link $n23 $n25 1.15Mb 1.5ms DropTail
$ns duplex-link $n23 $n26 1.15Mb 1.5ms DropTail
$ns duplex-link $n23 $n27 1.15Mb 1.5ms DropTail
$ns duplex-link $n23 $n28 1.15Mb 1.5ms DropTail
$ns duplex-link $n23 $n29 1.15Mb 1.5ms DropTail
$ns duplex-link $n23 $n30 1.15Mb 1.5ms DropTail

#La cola maxima entre los nodos $n0 y $n1 sera de 20 paquetes, el resto sera descartado
$ns queue-limit $n0 $n7 10
$ns queue-limit $n1 $n7 10
$ns queue-limit $n2 $n7 10
$ns queue-limit $n3 $n7 10
$ns queue-limit $n4 $n7 10
$ns queue-limit $n5 $n7 10
$ns queue-limit $n6 $n7 10
$ns queue-limit $n7 $n8 90
$ns queue-limit $n8 $n9 10
$ns queue-limit $n8 $n10 10
$ns queue-limit $n8 $n11 10
$ns queue-limit $n8 $n12 10
$ns queue-limit $n8 $n13 90
$ns queue-limit $n13 $n14 10
$ns queue-limit $n13 $n15 10
$ns queue-limit $n13 $n16 10
$ns queue-limit $n13 $n17 10
$ns queue-limit $n13 $n18 90
$ns queue-limit $n18 $n19 10
$ns queue-limit $n18 $n20 10
```

```
$ns queue-limit $n18 $n21 10
$ns queue-limit $n18 $n22 10
$ns queue-limit $n18 $n23 90
$ns queue-limit $n23 $n24 10
$ns queue-limit $n23 $n25 10
$ns queue-limit $n23 $n26 10
$ns queue-limit $n23 $n27 10
$ns queue-limit $n23 $n28 10
$ns queue-limit $n23 $n29 10
$ns queue-limit $n23 $n30 10

#Esto es opcional, es para dar la posicion de los nodos
#$ns duplex-link-op $n0 $n7 orient right-down
#$ns duplex-link-op $n1 $n7 orient right-down
#$ns duplex-link-op $n2 $n7 orient right-down
#$ns duplex-link-op $n3 $n7 orient right
#$ns duplex-link-op $n4 $n7 orient right-up
#$ns duplex-link-op $n5 $n7 orient right-up
#$ns duplex-link-op $n6 $n7 orient right-up

#--------------------------------------------
#Configuracion del agente TCP al nodo $n0
set tcp0 [new Agent/TCP]
#Agregar el agente al nodo $n0
$ns attach-agent $n0 $tcp0
#maximo tamaño del paquete
$tcp0 set packetSize_ 64

#Configuracion del agente SINK al nodo $n7
set ack0 [new Agent/TCPSink]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $ack0

#Conexion para los agentes tcp y sink
$ns connect $tcp0 $ack0

#Configuracion de la aplicacion
set ftp0 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp0 attach-agent $tcp0
#--------------------------------------------

#--------------------------------------------
#Configuracion del agente TCP al nodo $n1
set tcp1 [new Agent/TCP]
```

```
#Agregar el agente al nodo $n1
$ns attach-agent $n1 $tcp1
#maximo tamaño del paquete
$tcp1 set packetSize_ 64

#Configuracion del agente SINK al nodo $n7
set ack1 [new Agent/TCPSink]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $ack1

#Conexion para los agentes tcp y sink
$ns connect $tcp1 $ack1

#Configuracion de la aplicacion
set ftp1 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp1 attach-agent $tcp1
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n2
set tcp2 [new Agent/TCP]
#Agregar el agente al nodo $n2
$ns attach-agent $n2 $tcp2
#maximo tamaño del paquete
$tcp2 set packetSize_ 64

#Configuracion del agente SINK al nodo $n7
set ack2 [new Agent/TCPSink]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $ack2

#Conexion para los agentes tcp y sink
$ns connect $tcp2 $ack2

#Configuracion de la aplicacion
set ftp2 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp2 attach-agent $tcp2
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n3
set tcp3 [new Agent/TCP]
#Agregar el agente al nodo $n3
```

```
$ns attach-agent $n3 $tcp3
#maximo tamaño del paquete
$tcp3 set packetSize_ 64

#Configuracion del agente SINK al nodo $n7
set ack3 [new Agent/TCPSink]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $ack3

#Conexion para los agentes tcp y sink
$ns connect $tcp3 $ack3

#Configuracion de la aplicacion
set ftp3 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp3 attach-agent $tcp3
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n4
set tcp4 [new Agent/TCP]
#Agregar el agente al nodo $n4
$ns attach-agent $n4 $tcp4
#maximo tamaño del paquete
$tcp4 set packetSize_ 64

#Configuracion del agente SINK al nodo $n7
set ack4 [new Agent/TCPSink]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $ack4

#Conexion para los agentes tcp y sink
$ns connect $tcp4 $ack4

#Configuracion de la aplicacion
set ftp4 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp4 attach-agent $tcp4
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n5
set tcp5 [new Agent/TCP]
#Agregar el agente al nodo $n5
$ns attach-agent $n5 $tcp5
```

```
#maximo tamaño del paquete
$tcp5 set packetSize_ 64

#Configuracion del agente SINK al nodo $n7
set ack5 [new Agent/TCPSink]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $ack5

#Conexion para los agentes tcp y sink
$ns connect $tcp5 $ack5

#Configuracion de la aplicacion
set ftp5 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp5 attach-agent $tcp5
#----------------------------------------

#----------------------------------------
#Configuracion del agente UDP al nodo $n6
set udp6 [new Agent/UDP]
#Agregar el agente al nodo $n6
$ns attach-agent $n6 $udp6

#Configuracion del agente NULL al nodo $n7
set null6 [new Agent/Null]
#Agregar el agente al nodo $n7
$ns attach-agent $n7 $null6

#Conexion para los agentes UDP y null
$ns connect $udp6 $null6

#Configuracion de la aplicacion de UDP (CBR)
set cbr6 [new Application/Traffic/CBR]
#Agregue la aplicacion al agente
$cbr6 attach-agent $udp6
$cbr6 set type_ CBR
$cbr6 set packet_size_ 64
$cbr6 set rate_ 1mb
$cbr6 set random_ false
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n7
set tcp7 [new Agent/TCP]
#Agregar el agente al nodo $n7
```

```
$ns attach-agent $n7 $tcp7
#maximo tamaño del paquete
$tcp7 set packetSize_ 64

#Configuracion del agente SINK al nodo $n8
set ack7 [new Agent/TCPSink]
#Agregar el agente al nodo $n8
$ns attach-agent $n8 $ack7

#Conexion para los agentes tcp y sink
$ns connect $tcp7 $ack7

#Configuracion de la aplicacion
set ftp7 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp7 attach-agent $tcp7
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n8
set tcp8 [new Agent/TCP]
#Agregar el agente al nodo $n8
$ns attach-agent $n8 $tcp8
#maximo tamaño del paquete
$tcp8 set packetSize_ 64

#Configuracion del agente SINK al nodo $n13
set ack8 [new Agent/TCPSink]
#Agregar el agente al nodo $n13
$ns attach-agent $n13 $ack8

#Conexion para los agentes tcp y sink
$ns connect $tcp8 $ack8

#Configuracion de la aplicacion
set ftp8 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp8 attach-agent $tcp8
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n9
set tcp9 [new Agent/TCP]
#Agregar el agente al nodo $n9
$ns attach-agent $n9 $tcp9
```

```
#maximo tamaño del paquete
$tcp9 set packetSize_ 64

#Configuracion del agente SINK al nodo $n8
set ack9 [new Agent/TCPSink]
#Agregar el agente al nodo $n8
$ns attach-agent $n8 $ack9

#Conexion para los agentes tcp y sink
$ns connect $tcp9 $ack9

#Configuracion de la aplicacion
set ftp9 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp9 attach-agent $tcp9
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n10
set tcp10 [new Agent/TCP]
#Agregar el agente al nodo $n10
$ns attach-agent $n10 $tcp10
#maximo tamaño del paquete
$tcp10 set packetSize_ 64

#Configuracion del agente SINK al nodo $n8
set ack10 [new Agent/TCPSink]
#Agregar el agente al nodo $n8
$ns attach-agent $n8 $ack10

#Conexion para los agentes tcp y sink
$ns connect $tcp10 $ack10

#Configuracion de la aplicacion
set ftp10 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp10 attach-agent $tcp10
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n11
set tcp11 [new Agent/TCP]
#Agregar el agente al nodo $n11
$ns attach-agent $n11 $tcp11
#maximo tamaño del paquete
```

```
$tcp11 set packetSize_ 64

#Configuracion del agente SINK al nodo $n8
set ack11 [new Agent/TCPSink]
#Agregar el agente al nodo $n8
$ns attach-agent $n8 $ack11

#Conexion para los agentes tcp y sink
$ns connect $tcp11 $ack11

#Configuracion de la aplicacion
set ftp11 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp11 attach-agent $tcp11
#----------------------------------------

#----------------------------------------
#Configuracion del agente UDP al nodo $n12
set udp12 [new Agent/UDP]
#Agregar el agente al nodo $n12
$ns attach-agent $n12 $udp12

#Configuracion del agente NULL al nodo $n8
set null12 [new Agent/Null]
#Agregar el agente al nodo $n8
$ns attach-agent $n8 $null12

#Conexion para los agentes UDP y null
$ns connect $udp12 $null12

#Configuracion de la aplicacion de UDP (CBR)
set cbr12 [new Application/Traffic/CBR]
#Agregue la aplicacion al agente
$cbr12 attach-agent $udp12
$cbr12 set type_ CBR
$cbr12 set packet_size_ 64
$cbr12 set rate_ 1mb
$cbr12 set random_ false
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n13
set tcp13 [new Agent/TCP]
#Agregar el agente al nodo $n13
$ns attach-agent $n13 $tcp13
```

```
#maximo tamaño del paquete
$tcp13 set packetSize_ 64

#Configuracion del agente SINK al nodo $n18
set ack13 [new Agent/TCPSink]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $ack13

#Conexion para los agentes tcp y sink
$ns connect $tcp13 $ack13

#Configuracion de la aplicacion
set ftp13 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp13 attach-agent $tcp13
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n14
set tcp14 [new Agent/TCP]
#Agregar el agente al nodo $n14
$ns attach-agent $n14 $tcp14
#maximo tamaño del paquete
$tcp14 set packetSize_ 64

#Configuracion del agente SINK al nodo $n13
set ack14 [new Agent/TCPSink]
#Agregar el agente al nodo $n13
$ns attach-agent $n13 $ack14

#Conexion para los agentes tcp y sink
$ns connect $tcp14 $ack14

#Configuracion de la aplicacion
set ftp14 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp14 attach-agent $tcp14
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n15
set tcp15 [new Agent/TCP]
#Agregar el agente al nodo $n15
$ns attach-agent $n15 $tcp15
#maximo tamaño del paquete
```

```
$tcp15 set packetSize_ 64

#Configuracion del agente SINK al nodo $n13
set ack15 [new Agent/TCPSink]
#Agregar el agente al nodo $n13
$ns attach-agent $n13 $ack15

#Conexion para los agentes tcp y sink
$ns connect $tcp15 $ack15

#Configuracion de la aplicacion
set ftp15 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp15 attach-agent $tcp15
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n16
set tcp16 [new Agent/TCP]
#Agregar el agente al nodo $n16
$ns attach-agent $n16 $tcp16
#maximo tamaño del paquete
$tcp16 set packetSize_ 64

#Configuracion del agente SINK al nodo $n13
set ack16 [new Agent/TCPSink]
#Agregar el agente al nodo $n13
$ns attach-agent $n13 $ack16

#Conexion para los agentes tcp y sink
$ns connect $tcp16 $ack16

#Configuracion de la aplicacion
set ftp16 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp16 attach-agent $tcp16
#----------------------------------------

#----------------------------------------
#Configuracion del agente UDP al nodo $n17
set udp17 [new Agent/UDP]
#Agregar el agente al nodo $n17
$ns attach-agent $n17 $udp17

#Configuracion del agente NULL al nodo $n13
```

```
set null17 [new Agent/Null]
#Agregar el agente al nodo $n13
$ns attach-agent $n13 $null17

#Conexion para los agentes UDP y null
$ns connect $udp17 $null17

#Configuracion de la aplicacion de UDP (CBR)
set cbr17 [new Application/Traffic/CBR]
#Agregue la aplicacion al agente
$cbr17 attach-agent $udp17
$cbr17 set type_ CBR
$cbr17 set packet_size_ 64
$cbr17 set rate_ 1mb
$cbr17 set random_ false
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n18
set tcp18 [new Agent/TCP]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $tcp18
#maximo tamaño del paquete
$tcp18 set packetSize_ 64

#Configuracion del agente SINK al nodo $n23
set ack18 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack18

#Conexion para los agentes tcp y sink
$ns connect $tcp18 $ack18

#Configuracion de la aplicacion
set ftp18 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp18 attach-agent $tcp18
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n19
set tcp19 [new Agent/TCP]
#Agregar el agente al nodo $n19
$ns attach-agent $n19 $tcp19
#maximo tamaño del paquete
```

```
$tcp19 set packetSize_ 64

#Configuracion del agente SINK al nodo $n18
set ack19 [new Agent/TCPSink]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $ack19

#Conexion para los agentes tcp y sink
$ns connect $tcp19 $ack19

#Configuracion de la aplicacion
set ftp19 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp19 attach-agent $tcp19
#-----------------------------------------

#-----------------------------------------
#Configuracion del agente TCP al nodo $n20
set tcp20 [new Agent/TCP]
#Agregar el agente al nodo $n20
$ns attach-agent $n20 $tcp20
#maximo tamaño del paquete
$tcp20 set packetSize_ 64

#Configuracion del agente SINK al nodo $n18
set ack20 [new Agent/TCPSink]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $ack20

#Conexion para los agentes tcp y sink
$ns connect $tcp20 $ack20

#Configuracion de la aplicacion
set ftp20 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp20 attach-agent $tcp20
#-----------------------------------------

#-----------------------------------------
#Configuracion del agente TCP al nodo $n21
set tcp21 [new Agent/TCP]
#Agregar el agente al nodo $n21
$ns attach-agent $n21 $tcp21
#maximo tamaño del paquete
$tcp21 set packetSize_ 64
```

```
#Configuracion del agente SINK al nodo $n18
set ack21 [new Agent/TCPSink]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $ack21

#Conexion para los agentes tcp y sink
$ns connect $tcp21 $ack21

#Configuracion de la aplicacion
set ftp21 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp21 attach-agent $tcp21
#-----------------------------------------

#-----------------------------------------
#Configuracion del agente UDP al nodo $n22
set udp22 [new Agent/UDP]
#Agregar el agente al nodo $n22
$ns attach-agent $n22 $udp22

#Configuracion del agente NULL al nodo $n18
set null22 [new Agent/Null]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $null22

#Conexion para los agentes UDP y null
$ns connect $udp22 $null22

#Configuracion de la aplicacion de UDP (CBR)
set cbr22 [new Application/Traffic/CBR]
#Agregue la aplicacion al agente
$cbr22 attach-agent $udp22
$cbr22 set type_ CBR
$cbr22 set packet_size_ 64
$cbr22 set rate_ 1mb
$cbr22 set random_ false
#-----------------------------------------

#-----------------------------------------
#Configuracion del agente TCP al nodo $n23
set tcp23 [new Agent/TCP]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $tcp23
#maximo tamaño del paquete
```

```
$tcp23 set packetSize_ 64

#Configuracion del agente SINK al nodo $n18
set ack23 [new Agent/TCPSink]
#Agregar el agente al nodo $n18
$ns attach-agent $n18 $ack23

#Conexion para los agentes tcp y sink
$ns connect $tcp23 $ack23

#Configuracion de la aplicacion
set ftp23 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp23 attach-agent $tcp23
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n24
set tcp24 [new Agent/TCP]
#Agregar el agente al nodo $n24
$ns attach-agent $n24 $tcp24
#maximo tamaño del paquete
$tcp24 set packetSize_ 64

#Configuracion del agente SINK al nodo $n23
set ack24 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack24

#Conexion para los agentes tcp y sink
$ns connect $tcp24 $ack24

#Configuracion de la aplicacion
set ftp24 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp24 attach-agent $tcp24
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n25
set tcp25 [new Agent/TCP]
#Agregar el agente al nodo $n25
$ns attach-agent $n25 $tcp25
#maximo tamaño del paquete
$tcp25 set packetSize_ 64
```

```
#Configuracion del agente SINK al nodo $n23
set ack25 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack25

#Conexion para los agentes tcp y sink
$ns connect $tcp25 $ack25

#Configuracion de la aplicacion
set ftp25 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp25 attach-agent $tcp25
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n26
set tcp26 [new Agent/TCP]
#Agregar el agente al nodo $n26
$ns attach-agent $n26 $tcp26
#maximo tamaño del paquete
$tcp26 set packetSize_ 64

#Configuracion del agente SINK al nodo $n23
set ack26 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack26

#Conexion para los agentes tcp y sink
$ns connect $tcp26 $ack26

#Configuracion de la aplicacion
set ftp26 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp26 attach-agent $tcp26
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n27
set tcp27 [new Agent/TCP]
#Agregar el agente al nodo $n27
$ns attach-agent $n27 $tcp27
#maximo tamaño del paquete
$tcp27 set packetSize_ 64
```

```
#Configuracion del agente SINK al nodo $n23
set ack27 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack27

#Conexion para los agentes tcp y sink
$ns connect $tcp27 $ack27

#Configuracion de la aplicacion
set ftp27 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp27 attach-agent $tcp27
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n28
set tcp28 [new Agent/TCP]
#Agregar el agente al nodo $n28
$ns attach-agent $n28 $tcp28
#maximo tamaño del paquete
$tcp28 set packetSize_ 64

#Configuracion del agente SINK al nodo $n23
set ack28 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack28

#Conexion para los agentes tcp y sink
$ns connect $tcp28 $ack28

#Configuracion de la aplicacion
set ftp28 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp28 attach-agent $tcp28
#----------------------------------------

#----------------------------------------
#Configuracion del agente TCP al nodo $n29
set tcp29 [new Agent/TCP]
#Agregar el agente al nodo $n29
$ns attach-agent $n29 $tcp29
#maximo tamaño del paquete
$tcp29 set packetSize_ 64

#Configuracion del agente SINK al nodo $n23
```

```
set ack29 [new Agent/TCPSink]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $ack29

#Conexion para los agentes tcp y sink
$ns connect $tcp29 $ack29

#Configuracion de la aplicacion
set ftp29 [new Application/FTP]
#Agregue la aplicacion al agente
$ftp29 attach-agent $tcp29
#----------------------------------------

#----------------------------------------
#Configuracion del agente UDP al nodo $n30
set udp30 [new Agent/UDP]
#Agregar el agente al nodo $n30
$ns attach-agent $n30 $udp30

#Configuracion del agente NULL al nodo $n23
set null30 [new Agent/Null]
#Agregar el agente al nodo $n23
$ns attach-agent $n23 $null30

#Conexion para los agentes UDP y null
$ns connect $udp30 $null30

#Configuracion de la aplicacion de UDP (CBR)
set cbr30 [new Application/Traffic/CBR]
#Agregue la aplicacion al agente
$cbr30 attach-agent $udp30
$cbr30 set type_ CBR
$cbr30 set packet_size_ 64
$cbr30 set rate_ 1mb
$cbr30 set random_ false
#----------------------------------------

#Programacion de eventos, por ejemplo, $cbr comienza a los 0.1 segundos y termina a los 5 segundos
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "$ftp1 start"
$ns at 0.1 "$ftp2 start"
$ns at 0.1 "$ftp3 start"
$ns at 0.1 "$ftp4 start"
$ns at 0.1 "$ftp5 start"
```

```
$ns at 0.1 "$cbr6 start"
$ns at 0.1 "$ftp7 start"
$ns at 0.1 "$ftp8 start"
$ns at 0.1 "$ftp9 start"
$ns at 0.1 "$ftp10 start"
$ns at 0.1 "$ftp11 start"
$ns at 0.1 "$cbr12 start"
$ns at 0.1 "$ftp13 start"
$ns at 0.1 "$ftp14 start"
$ns at 0.1 "$ftp15 start"
$ns at 0.1 "$ftp16 start"
$ns at 0.1 "$cbr17 start"
$ns at 0.1 "$ftp18 start"
$ns at 0.1 "$ftp19 start"
$ns at 0.1 "$ftp20 start"
$ns at 0.1 "$ftp21 start"
$ns at 0.1 "$cbr22 start"
$ns at 0.1 "$ftp23 start"
$ns at 0.1 "$ftp24 start"
$ns at 0.1 "$ftp25 start"
$ns at 0.1 "$ftp26 start"
$ns at 0.1 "$ftp27 start"
$ns at 0.1 "$ftp28 start"
$ns at 0.1 "$ftp29 start"
$ns at 0.1 "$cbr30 start"
$ns at 4.9 "$ftp0 stop"
$ns at 4.9 "$ftp1 stop"
$ns at 4.9 "$ftp2 stop"
$ns at 4.9 "$ftp3 stop"
$ns at 4.9 "$ftp4 stop"
$ns at 4.9 "$ftp5 stop"
$ns at 4.9 "$cbr6 stop"
$ns at 4.9 "$ftp7 stop"
$ns at 4.9 "$ftp8 stop"
$ns at 4.9 "$ftp9 stop"
$ns at 4.9 "$ftp10 stop"
$ns at 4.9 "$ftp11 stop"
$ns at 4.9 "$cbr12 stop"
$ns at 4.9 "$ftp13 stop"
$ns at 4.9 "$ftp14 stop"
$ns at 4.9 "$ftp15 stop"
$ns at 4.9 "$ftp16 stop"
$ns at 4.9 "$cbr17 stop"
$ns at 4.9 "$ftp18 stop"
$ns at 4.9 "$ftp19 stop"
```

```
        $ns at 4.9 "$ftp20 stop"
        $ns at 4.9 "$ftp21 stop"
        $ns at 4.9 "$cbr22 stop"
        $ns at 4.9 "$ftp23 stop"
        $ns at 4.9 "$ftp24 stop"
        $ns at 4.9 "$ftp25 stop"
        $ns at 4.9 "$ftp26 stop"
        $ns at 4.9 "$ftp27 stop"
        $ns at 4.9 "$ftp28 stop"
        $ns at 4.9 "$ftp29 stop"
        $ns at 4.9 "$cbr30 stop"

        # Hasta aca se inserta el codigo personalizado

        #Al pasar los 5 segundos, para finalizar, llamar a la funcion 'finish'
        $ns at 5.0 "finish"

        #Iniciar la simulacion
$ns run
```