EVALUACIÓN DE HERRAMIENTAS DE ESTIMACIÓN DE ANCHO DE BANDA DISPONIBLE EN UNA RED DE COMPUTADORAS

YOEL LÓPEZ CASTELL

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
TELECOMUNICACIONES Y TECNOLOGÍA WEB
BUCARAMANGA, 2010

EVALUACIÓN DE HERRAMIENTAS DE ESTIMACIÓN DE ANCHO DE BANDA DISPONIBLE EN UNA RED DE COMPUTADORAS

YOEL LÓPEZ CASTELL

Trabajo de grado presentado para optar por el título de ingeniero de sistemas

Director:
PhD. César D. Guerrero Santander

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
TELECOMUNICACIONES Y TECNOLOGÍA WEB
BUCARAMANGA, 2010

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ECUACIONES

pág.

RESUMEN

Diferentes protocolos y aplicaciones en el internet pueden usar información del ancho de banda disponible en la red para mejorar su rendimiento. Existen diferentes estimadores de ancho de banda cuyo desempeño depende del método de estimación empleado y del entorno de red donde se aplique. A través de esta investigación se evaluó la operación de los principales estimadores de ancho de banda disponible.

Para hacer las evaluaciones de los estimadores se seleccionaron los más importantes según los resultados obtenidos en el estado del arte y se creó una infraestructura de red controlada (*testbed*) que emula el comportamiento del internet. Se seleccionaron siete herramientas de estimación: *Spruce*, *Traceband* y *IGI* (que usan un modelo llamado "*Probe Gap Model*"), y *Pathload*, *Pathchirp*, *PTR* y *Yaz* (que usan un modelo llamado "*Probe Rate Model*"). Se seleccionaron 10 escenarios de red diferentes utilizando la herramienta de generación de tráfico sintético llamada *Mgen*.

El proyecto se enmarca en la línea de investigación en telemática del grupo de investigación en tecnologías de información (GTI). Además de la contribución de la investigación y evaluación realizada, éste proyecto proporcionó a la línea y al semillero de investigación de la misma una infraestructura (*testbed*) que potencia el desarrollo de proyectos adicionales en las líneas que se desarrollen en el marco de programas de pregrado y de posgrado de la UNAB.

Palabras Claves

Estimación de ancho de banda, Testbed, Generación de tráfico sintético

INTRODUCCIÓN

Este proyecto enmarcado dentro de la línea de investigación en Telemática del grupo de investigación en tecnologías de la información de la UNAB, se planteó como objetivo el realizar una evaluación de las principales herramientas de estimación de ancho de banda disponible, en un escenario controlado que emule el comportamiento de una red en la internet. Para esto, se creó una infraestructura de red que permitió evaluar estimadores de ancho de banda disponibles simulando el comportamiento del internet (*testbed*). En dicha infraestructura, se evaluaron las principales herramientas de estimación de ancho de banda disponible (*Igi*, *Ptr*, *Spruce*, *Traceband*, *Pathload*, *Pathchrip* y *Yaz*) utilizando *Mgen* para la generación de tráfico sintético.

El estimar el ancho de banda disponible es un problema estudiado a nivel mundial por investigadores dada la necesidad de contar con esta información para mejorar las operaciones de varias aplicaciones de red, tales como el cumplimiento de los acuerdos de nivel de servicio, gestión de redes, ingeniería de tráfico y en tiempo real de los recursos de aprovisionamiento, control de flujo y congestión, la detección rápida de fallas, ataques de red y control de admisión[1]. Evaluar las herramientas de estimación de ancho de banda disponible permite determinar cuáles estimadores son más adecuados para ser utilizados por diferentes aplicaciones en diferentes escenarios de red.

[1] Guerrero Cesar, Labrador Miguel, Traceband: A fast, low overhead and accurate tool for available bandwidth estimation and monitoring. Internet Network Management Workshop (INM 2008). October 2008. Orlando, FL.

Existen estudios previos que evalúan herramientas de estimación. La principal contribución de este proyecto respecto a otras evaluaciones como la realizada por el director del proyecto, radica en la inclusión de las más recientes herramientas de estimación como *Yaz* y *Traceband* en un escenario de red totalmente controlado.

1. ANCHO DE BANDA DISPONIBLE EN LA RED

1.1 ANCHO DE BANDA DISPONIBLE

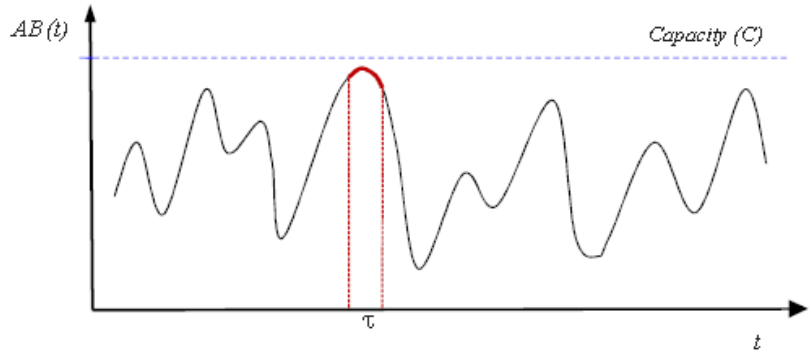
1.1.1 Definición En las redes informáticas, el ancho de banda es una medida de velocidad que se define como la cantidad de bits transmitidos en un canal de comunicaciones por unidad de tiempo. Generalmente se especifica en bit por segundo (bps). Hay dos métricas relacionadas con el ancho de banda, una es la capacidad y la otra es el ancho de banda disponible. La capacidad de un enlace es la cantidad máxima de bits que se pueden transmitir al enlace por unidad de tiempo, éste es el ancho de banda máximo.

1.1.2 Ancho de banda disponible de extremo a extremo El mínimo de todas las capacidades de enlace no utilizadas en toda la ruta de comunicación se llama *ancho de banda disponible de extremo a extremo*. Ésta es una métrica variable en el tiempo que está relacionada con la utilización individual de cada enlace en toda la ruta. Definiendo a τ como la escala de tiempo promedio (Ver Figura 1) del ancho de banda disponible, la utilización promedio del enlace *i* para una muestra de tiempo τ, se obtiene por medio de la Ecuación 1.

Ecuación 1. Formula del ancho de banda promedio disponible.

$$\overline{u_i} = \frac{1}{\tau}\int_t^{t+\tau} u_i(s)ds, \quad 0 \leq \overline{u_i} \leq 1.$$

Figura 1. Ancho de banda disponible en un período promedio de la escala de tiempo



Fuente: Guerrero Santander Cesar, 2009, End-to-End Available Bandwidth Estimation and Monitoring. ISBN: 9781109621884

Para un enlace *i* con una capacidad *Ci*, el ancho de banda disponible del enlace en el intervalo (t, t + τ) se puede definir como la capacidad no utilizada promedio durante el tiempo τ (ver Ecuación 2).

Ecuación 2. Capacidad no utilizada promedio.

$$\overline{A_i} = C_i\left[1 - \overline{u_i}\right].$$

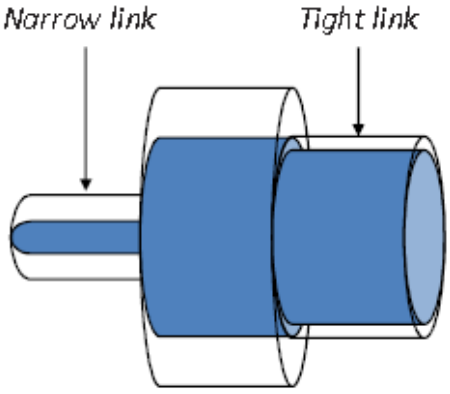
Para una ruta de extremo a extremo con *H* saltos, el ancho de banda disponible durante τ lo da el enlace con la capacidad no utilizada mínima de todos los saltos, (ver Ecuación 3).

Ecuación 3. Capacidad no utilizada mínima.

$$\overline{A} = min_{i=1..H}(\overline{A_i}).$$

Tal como aparece en la Ecuación 2, el enlace con la capacidad mínima se conoce como *narrow link* (enlace angosto) y el enlace con el mínimo ancho de banda disponible se conoce como *tight link* (enlace estrecho) (ver Figura 2), el cual se considera el cuello de botella de la ruta y a su vez determina el ancho de banda disponible de extremo a extremo.

Figura 2. Angosto y apretado (Narrow and tight links)



Fuente: Guerrero Santander Cesar, 2009, End-to-End Available Bandwidth Estimation and Monitoring. ISBN: 9781109621884

2. TÉCNICAS DE ESTIMACIÓN DE ANCHO DE BANDA Y HERRAMIENTAS DISPONIBLES.

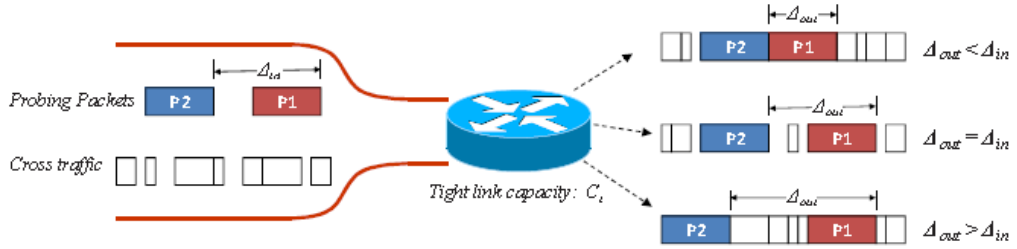
Hay dos enfoques diferentes para medir el ancho de banda disponible en una ruta de extremo a extremo: *probe gap model* (*PGM*) (modelo de separación de pruebas) y *probe rate model* (*PRM*) (modelo de velocidad de prueba). El *PGM* observa la dispersión del par de paquetes de prueba, mientras que el *PRM* observa los retardos en un sentido de los paquetes de prueba.

Ambos enfoques utilizan un tren de paquetes de prueba para generar una medición promediada y así superar la naturaleza de variabilidad abrupta del tráfico cruzado.

Aunque no es necesario para que funcionen las herramientas de medición, se requiere que éstas sostengan dos suposiciones para la validez analítica de los modelos de medición:

- Los enrutadores a lo largo de la ruta presentan una disciplina de cola FIFO.
- El modelo de enlace sencillo que aparece en la Figura 3 es uno donde la velocidad del tráfico cruzado es constante durante la escala de tiempo promedio τ.

Figura 3. Modelo de enlace sencillo para la medición de ancho de banda.



Tomado de: Guerrero Santander Cesar, 2009, End-to-End Available Bandwidth Estimation and Monitoring. ISBN: 9781109621884

2.1 PROBE GAP MODEL

Éste modelo basa la medición en la dispersión de espacios (*gap dispersion*) entre dos paquetes de prueba consecutivos en el receptor, lo cual tiene una fuerte correlación con la cantidad de tráfico cruzado en el enlace estrecho. La dispersión aumenta linealmente con la velocidad del tráfico cruzado si la cola del enlace estrecho (Figura 3) no se desocupa después que el primer paquete del par sale del enrutador y antes que el segundo paquete llegue al enrutador[2]. Por lo tanto, el ancho de banda disponible se mide determinando la cantidad de tráfico cruzado y restándolo de la capacidad conocida del enlace estrecho (ver Ecuación 4) donde ε es la dispersión relativa o *strain* (deformación) definida en la Ecuación 5.

Ecuación 4. Capacidad del enlace estrecho.

$$\overline{A} = C \times (1 - \varepsilon)$$

[2] Michaut Fabien, Lepage Francis, CRAN CNRS UMR7039, Application-Oriented Network Metrology: Metrics And Actives Measurement Tools.

Ecuación 5. Dispersión relativa.

$$\varepsilon = \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}.$$

Los ejemplos de herramientas de medición disponibles basadas en el enfoque del modelo de separación de pruebas son *Traceband* [3], *Spruce*[4] y *IGI*[5]

2.2 PROBE RATE MODEL

Éste es un modelo basado en la idea de congestión inducida, donde se envían trenes de paquetes de prueba a velocidades cada vez mayores y el receptor observa las variaciones en el retardo promedio en un sentido del tren, buscándose así el punto de inflexión o el punto en el cual el retardo del paquete de prueba empieza a aumentar de manera consistente. Si un tren se envía a una velocidad menor al ancho de banda disponible de la ruta, el tren experimentará retardos similares.

De lo contrario, si la velocidad del tren es mayor al ancho de banda disponible de la ruta, el tren hará cola en el enrutador del enlace estrecho y experimentará retardos cada vez mayores (punto de cambio). Entonces se mide el ancho de banda disponible mirando la velocidad del paquete de prueba utilizada cuando se

[3] Guerrero Cesar, Labrador Miguel, Traceband: A fast, low overhead and accurate tool for available bandwidth estimation and monitoring. Internet Network Management Workshop (INM 2008). October 2008. Orlando, FL.
[4] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of the 3rd ACM SIGCOMM conference on Internet Measurement, 2003, pp. 39–44.
[5] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal on Selected Areas in Communications 21 (6) (2003) 879–894.

encontró el punto de inflexión o cambio. En éste punto, la velocidad del tren es igual al ancho de banda disponible en la ruta de extremo a extremo. Los ejemplos de herramientas en el modelo de la velocidad de prueba son *Pathload*[6] , *Pathchirp*[7] y *Yaz*.

Éste método se conoció inicialmente como el mecanismo del tren de par de paquetes (*TOPP*) (*train of packet pair*) como así lo definió *Melander*[8]. Él propuso inyectar pares de paquetes de prueba en la red y observar en el receptor los tiempos de recepción de los paquetes de prueba para medir el ancho de banda disponible.

El emisor comienza a transmitir un conjunto de *n* pares separados de paquetes de igual tamaño *L* a una velocidad *Rmin*. Luego se aumenta esta velocidad y se envía otro tren. Esto continúa hasta que se llega a la velocidad de prueba máxima *Rmax*. El ancho de banda disponible se mide a partir de la relación entre las velocidades de entrada y salida. El *TOPP* sólo se simuló utilizando el simulador de red *ns-2*[9].

---

[6] M. Jain, C. Dovrolis, Pathload: A measurement tool for end-to-end available bandwidth, in: Proceedings of the 3rd Passive and Active Measurements Workshop, vol. 11, 2002, pp. 14–25.
[7] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, L. Cottrell, pathchirp: Efficient available bandwidth estimation for network paths, in: Proceedings of the 4th Passive and Active Measurements Workshop, vol. 2, 2003.
[8] B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estimating bandwidth bottlenecks, in: Proceedings of the IEEE Global Telecommunications Conference, vol. 1, San Francisco, CA, USA, 2000, pp. 415–420.
[9] Ns-2 The Network Simulator NS 2, Home page. Página principal.

http://www.isi.edu/nsnam/ns/. Junio 2010.

2.3 HERRAMIENTAS DE MEDICIÓN DE ANCHO DE BANDA DISPONIBLE

Ésta sección describe las herramientas de medición como las presentan sus autores. La notación básica que se utiliza para explicar las operaciones de las herramientas se basan en el *modelo de enlace sencillo* que se presentó en la Sección 2.

2.3.1 Spruce *Spruce*[10] utiliza el enfoque del *modelo de separación de pruebas* para hacer sus mediciones. Envía un muestreo *Poisson* de pares de paquetes UDP de 1500 Bytes con una separación entre pares igual al tiempo de transmisión del *narrow link* de un paquete de 1500 Bytes. Eso garantiza que el segundo paquete llega a la cola del *narrow link* antes que el primer paquete salga de la cola. Al establecer el espacio entre pares según la salida de una función distribuida exponencialmente, *Spruce* lleva a cabo un proceso de muestreo *Poisson* que permite que la herramienta no sea intrusiva.

Al utilizar la dispersión de los paquetes de prueba medida en el receptor, *Spruce* calcula la razón promedio del tráfico que llega a la cola entre los dos paquetes como la capacidad del *tight link Ct* multiplicado por la dispersión relativa obtenida de la Ecuación 5.

El ancho de banda disponible se determina restando esa razón de tráfico cruzado de la capacidad en el *tight link*. Después de hacer mediciones de prueba K, la herramienta reporta el promedio de todos los anchos de banda disponibles que se calcularon. El valor predeterminado para *K* es 100. La medición de *Spruce* requiere un cálculo previo de la capacidad del *tight link*.

[10] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of the 3rd ACM SIGCOMM conference on Internet Measurement, 2003, pp. 39–44.

2.3.2 Igi *Igi*[11] utiliza el modelo *PGM*. Los autores desarrollan dos técnicas de pares de paquetes para caracterizar el ancho de banda disponible. Una es *IGI* (initial gap increasing) y la otra *PTR* (packet transmission rate). Éstas técnicas se utilizan para determinar experimentalmente la separación inicial (Δ_{in}) que producirá una alta correlación entre el *throughput* (volumen de información) del tráfico competente en el *tight link* y el espacio de salida (Δ_{out}) en el destino.

IGI encuentra un valor de espacio de prueba inicial para que el tren de paquetes de prueba interactúe con el tráfico cruzado en una cola del *narrow link* no vacía, lo cual los autores llaman *Joint Queuing Region* (JQR). En esa región, hay una relación proporcional entre el espacio, cuando los paquetes de prueba salen de la cola (espacio de salida) y el tráfico cruzado.

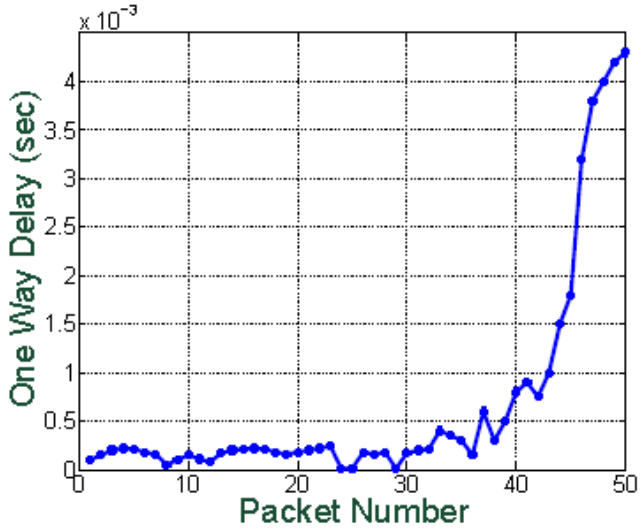
2.3.3 Pathload *Pathload*[12] [11] utiliza el *Self-Loading Periodic Stream* (SLoPS), una técnica que sigue el mismo principio del modelo *PRM*. En términos generales, *SLoPS* se basa en el hecho de que el retardo en un sentido de un *stream* de paquetes periódicos aumenta cuando la razón del tráfico de prueba es más alta que el ancho de banda disponible en la ruta. De lo contrario, no hay aumento en el retardo medido. Una *fleet* de *streams* (número fijo de paquetes) es enviado a razones variables y entonces la tendencia del retardo en un sentido de cada *stream* se caracteriza en el receptor como creciente o decreciente. Cuando ese retardo está en la región gris donde no hay una tendencia claramente creciente o decreciente (ver Figura 4), la metodología presenta un rango en el qué el ancho de banda disponible varía.

---

[11] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal on Selected Areas in Communications 21 (6)(2003) 879–894.
[12] M. Jain, C. Dovrolis, Pathload: A measurement tool for end-to-end available bandwidth, in: Proceedings of the 3rd Passive and Active Measurements Workshop, vol. 11, 2002, pp. 14–25

No hay una tendencia claramente creciente o decreciente en el retardo en un sentido entre los paquetes 35 y 40.

Figura 4. Región gris del Pathload.



Fuente: Guerrero Santander Cesar, 2009, End-to-End Available Bandwidth Estimation and Monitoring. ISBN: 9781109621884

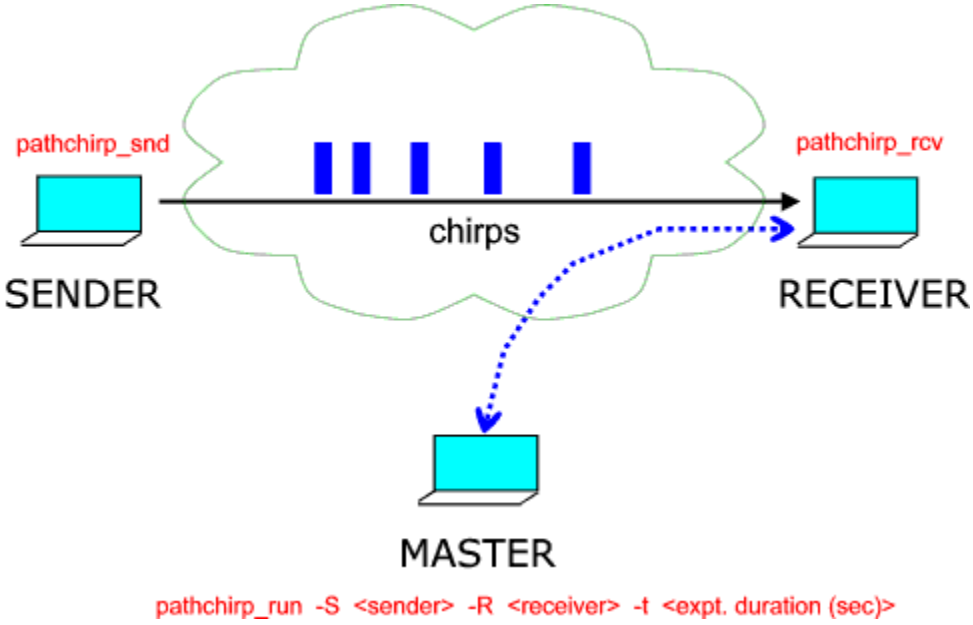
2.3.4 Traceband *Traceband* [13], es una herramienta cliente-servidor escrita en ANSI C, utiliza la representación oculta descrita de *Markov*[14], en la dinámica de ancho de banda disponible para proporcionar estimaciones rápidas, AB continua y precisa. El cliente que ejecuta *Traceband* cuenta con ciclos de diez estimaciones. En la primera estimación de la herramienta envía 50 pares de paquetes UDP 1498 Bytes de longitud. Las restantes nueve estimaciones se realizan con 30 pares de paquetes cada una.

[13] Guerrero Cesar, Labrador Miguel, Traceband: A fast, low overhead and accurate tool for available bandwidth estimation and monitoring. Internet Network Management Workshop (INM 2008). October 2008. Orlando, FL.
[14] Guerrero Cesar, Labrador Miguel, Traceband: A fast, low overhead and accurate tool for available bandwidth estimation and monitoring. Internet Network Management Workshop (INM 2008). October 2008. Orlando, FL.

Esta reducción es posible gracias a la *HMM*, que es capaz de aprender la dinámica AB con una muestra inicial y mantener el modelo actualizado con muestras de tamaño reducido. Se encontró en la experimentación de re-aprendizaje en cada diez estimaciones que es suficiente para mantener una buena precisión con bajo costo operativo. Usando 50 pares de prueba el modelo mostró los mejores resultados de estimación. Futuros trabajos estudiaran la implicación de la precisión de la herramienta cuando éste número de paquetes de sondeo varíe.

2.3.5 PathChirp *PatChirp* es una herramienta activa de sondeo para estimar el ancho de banda disponible en una ruta de red de comunicación. Se basa en el concepto de "auto-congestión inducida" *PRM*; cuenta con un patrón de vuelo exponencial en las sondas al que llaman *chirp*. Paquetes de *chirp* que ofrecen varias ventajas significativas sobre los actuales sistemas de sondeo, sobre la base de pares de paquetes o trenes de paquetes. Por el rápido aumento de la tasa de sondeo dentro de cada *chirp*, *PathChirp* obtiene un amplio conjunto de información para calcular dinámicamente el ancho de banda disponible.

Figura 5 Descripción del funcionamiento de PathChirp.



Tomado de: http://www.spin.rice.edu/Software/pathChirp/

3. TESTBED DE RED

3.1 DEFINICIÓN

Un banco de pruebas (*testbed*) es una plataforma para experimentación de proyectos de gran desarrollo. Los bancos de pruebas brindan una forma de comprobación rigurosa, transparente y repetible de teorías científicas, elementos computacionales y otras nuevas tecnologías.

El término se usa en varias disciplinas para describir un ambiente de desarrollo que está protegido de los riesgos de las pruebas en un ambiente de producción. Es un método para probar un módulo particular (función, clase o biblioteca) en forma aislada. Un banco de pruebas se usa cuando un nuevo módulo se prueba aparte del programa al que luego será agregado.

3.2 TESTBED UTILIZADOS PARA LA ESTIMACIÓN DE ANCHO DE NANDA

3.2.1 Planetlab *Planetlab* es un escenario de pruebas de dimensiones globales, ha sido diseñado para apoyar el desarrollo de nuevos servicios en redes académicas avanzadas. Nace en el 2003 liderado por la Universidad de Princeton en Estados Unidos, se construye gracias a la suma de un gran número de servidores distribuidos a través de las redes académicas del mundo, los que a su vez, forman un laboratorio computacional a escala planetaria; de ahí viene su nombre.

En el conjunto de servidores que componen la red de *PlanetLab* se pueden desarrollar, instalar y ejecutar aplicaciones en un entorno de prueba "*testbed*" desplegado sobre una red con condiciones del mundo-real. Más de 800 servidores repartidos en sobre 400 sitios de más de 40 países del mundo, albergan la implementación que posibilita la existencia de *PlanetLab* y donan parte de su ancho de banda para que este efectivamente logre operar.

La mayoría de sus servidores están instalados en universidades conectadas a las redes académicas y otros en los Centros de Operaciones de esas redes, como en el caso de los nodos de la red avanzada latinoamericana, RedCLARA (denominados "PlanetLab Colo - CLARA", y dispuestos en las ciudades de Buenos Aires, Ciudad de Panamá, Santiago - REUNA, Sao Paulo y Tijuana), Internet2 (Estados Unidos), etc.

3.2.2 ModelNet *ModelNet* es un emulador de redes a gran escala que permite a los usuarios evaluar los sistemas distribuidos en red en entornos realistas, como Internet. *ModelNet* permite el ensayo de prototipos sin modificar corriendo sobre sistemas operativos sin modificar a través de diversos escenarios de redes. En cierto modo, se combina la capacidad de repetición de la simulación con el realismo de despliegue en vivo.

La comunidad de usuarios *ModelNet* ha desplegado para ayudar en el diseño y prueba de redes nuevas de distribución de contenidos, sistemas *peer-to-peer*, los protocolos de capa de transporte, los conmutadores basados en los contenidos, procesadores distribuidos de flujo, sistemas de archivos distribuidos y herramientas de red de medición.

Los usuarios despliegan *ModelNet* en su *cluster* de área local. Cada instancia de la aplicación se ejecuta en un nodo virtual; los nodos múltiples *ModelNet* virtual a

través de un conjunto de equipos físicos que llamamos nodos borde. El sistema configura los nodos de borde para encaminar sus paquetes a través de un núcleo *ModelNet* (que consiste en una o más máquinas físicas). Este toma básicamente los paquetes de retardo del ancho de banda y especifica la pérdida de éstos en la topología de destino. *ModelNet* soporta emulación *hop-by-hop*, capturada de los efectos de la congestión del tráfico cruzado y dentro de la red.

3.2.3 Web100 *Web100* fue desarrollado por un equipo de Pittsburgh Supercomputing Center, para proporcionar una visión de las características de una conexión TCP para desarrolladores de aplicaciones y administradores de sistemas. El proyecto fue creado específicamente para desarrollar una interfaz de gestión avanzada para TCP y exponer así el funcionamiento interno de éste. Ha sido utilizada con gran éxito en la identificación y diagnóstico de los síntomas en la causa de problemas de rendimiento de la red.

Web100 proporciona los instrumentos y herramientas para estudiar y diagnosticar las variables TCP. Proporciona acceso en el kernel a las variables internas del protocolo TCP para la configuración y características del funcionamiento para la retroalimentación instantánea del rendimiento de TCP.

Web100 trabaja mediante la modificación de los archivos de una versión específica del kernel de Linux, con el fin de exponer los detalles de TCP que de otra manera estarían ocultos.

4. MONTAJE DE HERRAMIENTAS Y PROCESAMIENTO DE LA INFORMACIÓN

4.1 HERRAMIENTAS SELECCIONADAS

Se seleccionaron para la evaluación tres herramientas de cada metodología, *Spruce, Traceband y Igi-Ptr* (Probe gap model), y *Pathload, Pathchirp y Yaz* (Probe rate model). Para la evaluación de las mismas se seleccionaron 10 escenarios diferentes para cada una, 1 sin tráfico y 3 con tráfico sintético *Periodic*, *Poisson y Burst* al 30%, 50% y 70% respectivamente, con la herramienta *MGEN* como generador de tráfico sintético.

Una vez puesta a punto la infraestructura de red, se instalaron en las máquinas 192.168.0.2 y 192.168.1.2 (ver Figura 21 Testbed de red), las herramientas seleccionadas a ser evaluadas, de igual manera, en las máquinas 192.168.0.3 y 192.168.1.3 se instaló el generador de tráfico sintético *MGEN*.

Para el montaje de las herramientas se busco en diferentes sitios en internet en los que se puede encontrar fácilmente los archivos para su descarga y posterior montaje. En la dirección http://netlab-mn.unipv.it/avail-bw/#configurations, se encontraron casi todas las herramientas en su última versión. La única excepción es *Traceband* que puede ser descargada de http://www.cse.usf.edu/guerrerc/traceband/soft.htm. La herramienta generadora de tráfico pude ser también descargada de http://downloads.pf.itd.nrl.navy.mil/mgen/.

4.1.1 Instalación las herramientas Se encontró que la mayoría de las herramientas no cuentan con una guía para instalación ni posterior uso. *Igi-Ptr* y

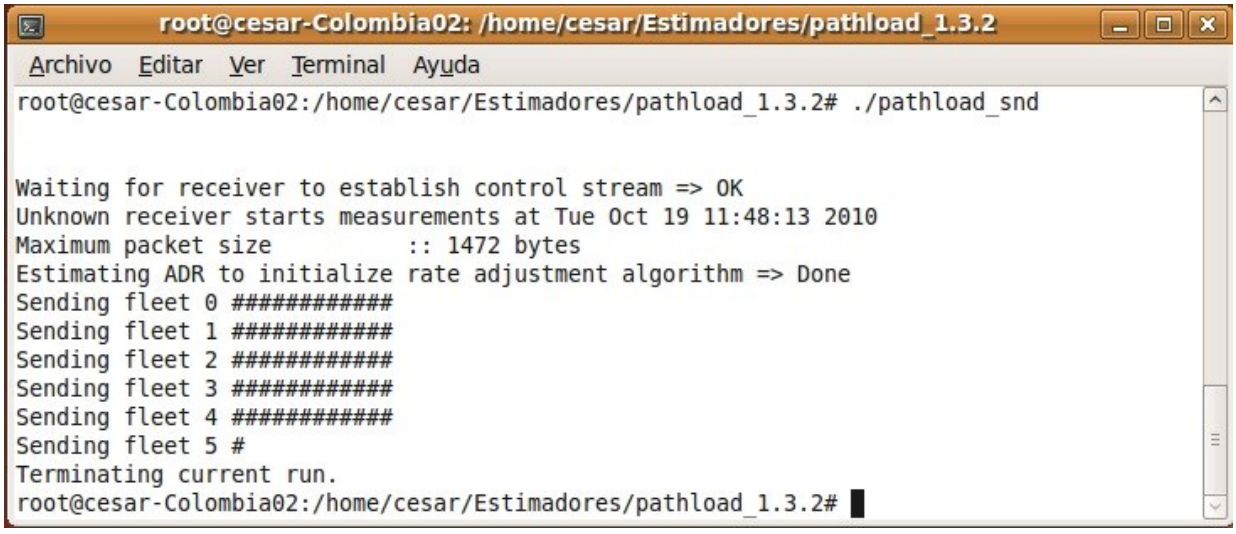
Pathchrip no poseen instructivos (como archivos README o INSTALL) que proporcionen detalles de instalación de las mismas. En las herramientas *Spruce*, *Yaz y Pathload* aunque si aparecen indicaciones de cómo instalarlas no cuentan con una guía de cómo hacer los experimentos con ellas correctamente después de instaladas. Los procedimientos de instalación de cada una de las herramientas son los siguientes:

Pathload

- Se descomprime el archivo descargado. Se accede a la carpeta donde quedó descomprimido a través de la consola y se ejecuta *./configure* y consecutivamente *make* para la creación de sus ejecutables.

- En la máquina cliente se compila el software de la siguiente manera: *#./pathload_snd*

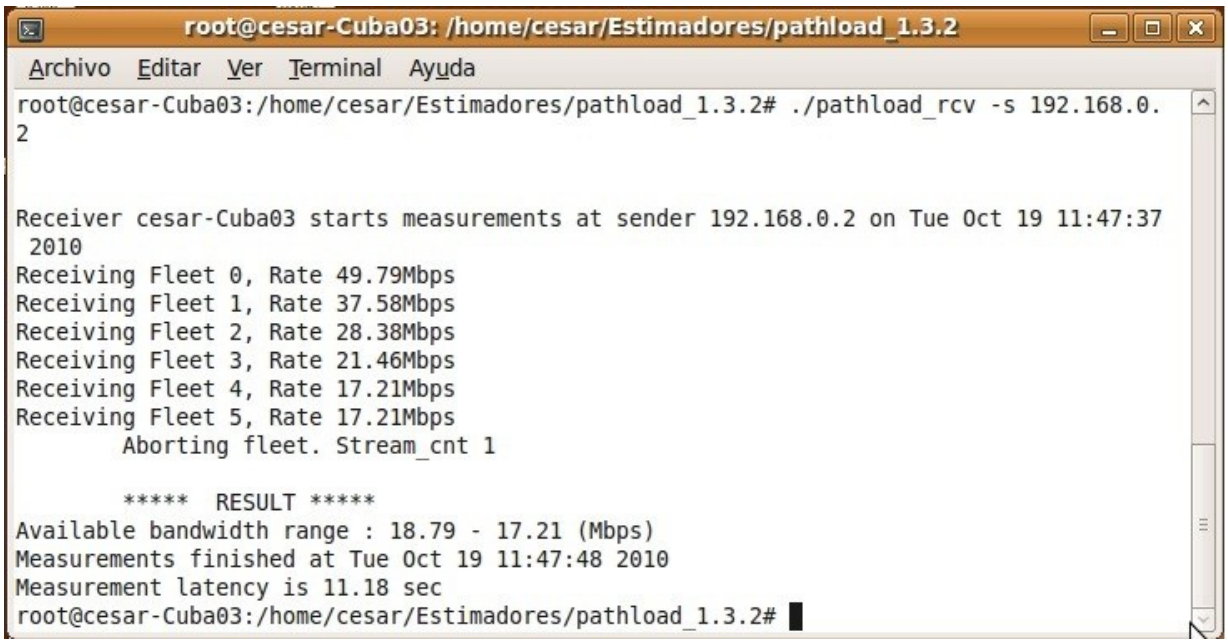
Figura 6 Pathload sender



Fuente: Autor del proyecto

- En la máquina receptor se compila el software de la siguiente manera: *#./pathload_rcv –s* [ipaddress]

Figura 7 Pathload reciver



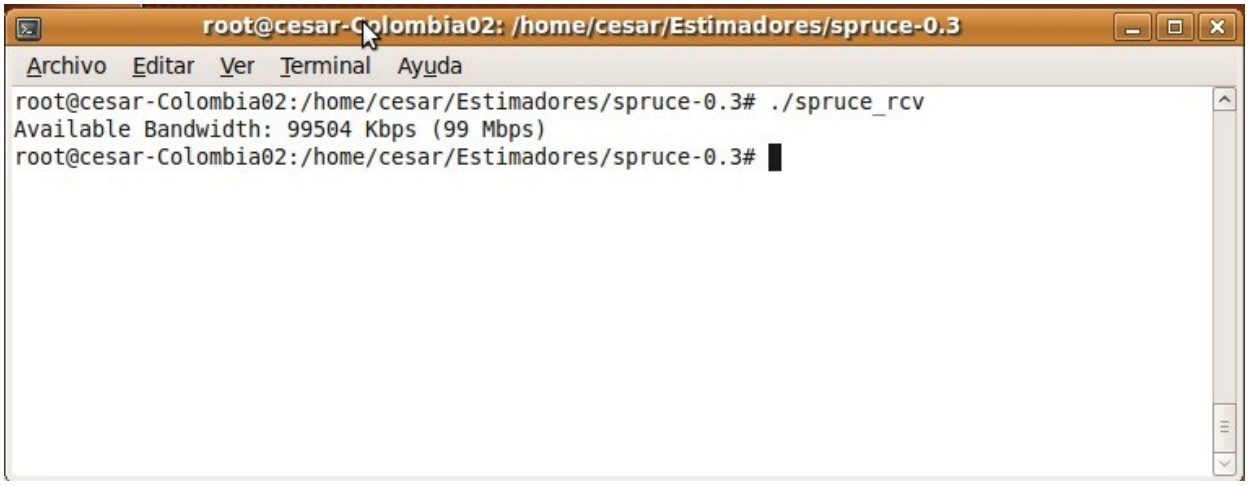
Fuente: Autor del proyecto

Spruce

- Se descomprime el archivo descargado. Se accede a la carpeta donde quedó descomprimido a través de la consola y se configura el archivo *Makefile*; se quita el comentario de la línea: CFLAGS = -g -Wall -O2 y se comenta la línea: #CFLAGS = -g -Wall -Werror -O2, ya que dan un error a la hora de a crear los ficheros de ejecución. Una vez hecho esto queda listo para que se puedan crear los ficheros de ejecución a través del comando *make*.

- En la máquina cliente se compila el software de la siguiente manera: *#./spruce_rcv*

Figura 8 Spruce reciver



Fuente: Autor del proyecto

- En la máquina receptor se compila el software de la siguiente manera: *#./spruce _snd –h* [ipaddress] –c [Path capacity]M

Figura 9 Spruce sender

Fuente: Autor del proyecto

Igi-Ptr

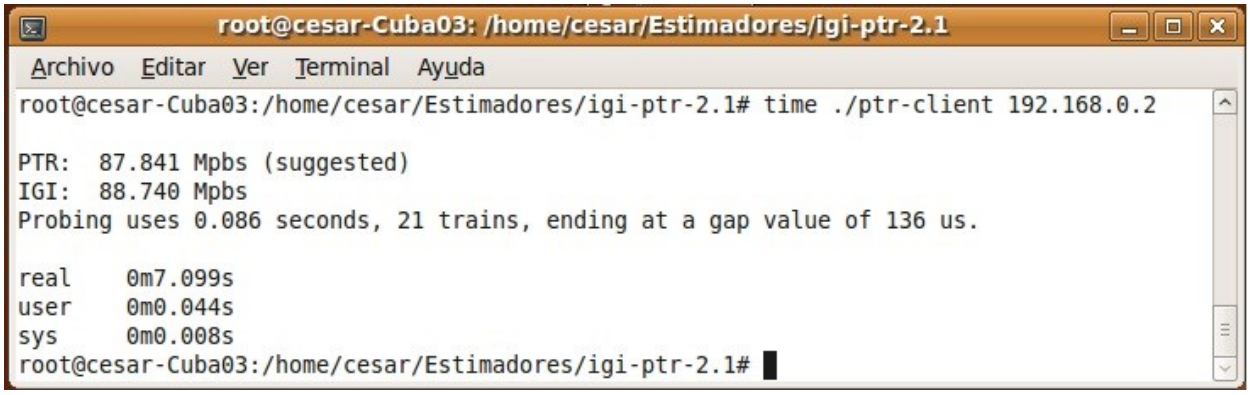
- Se descomprime el archivo descargado. Se accede a la carpeta donde quedó descomprimido a través de la consola y se ejecuta *./configure*, consecutivamente *make*.
- En la máquina cliente se compila el software de la siguiente manera: *#./ptr-server*

Figura 10 Ptr server

Fuente: Autor del proyecto

- En la máquina receptor se compila el software de la siguiente manera: *#./ptr-client* [ipaddress]

Figura 11 Ptr client



Fuente: Autor del proyecto

Pathchirp

- Se descomprime el archivo descargado. Se accede a la carpeta donde quedó descomprimido a través de la consola y se ejecuta: *./configure*, consecutivamente *make*.

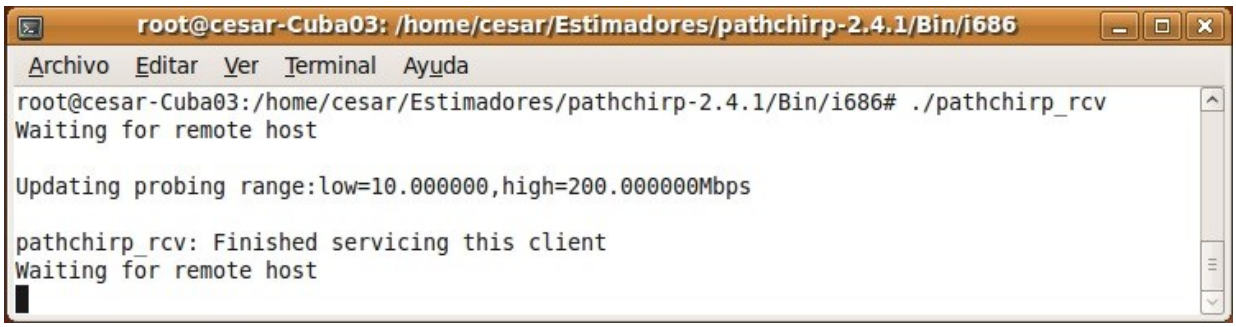
- En la máquina cliente se compila el software de la siguiente manera: *#./pathchrip_snd*

Figura 12 Pathchirp sender

Fuente: Autor del proyecto

- En la máquina receptor se compila el software de la siguiente manera: *#./pathchrip_rcv*

Figura 13 Pathchirp reciver



Fuente: Autor del proyecto

- Debe existir una tercera máquina para que se pueda compilar, pues ésta es la que ordena a amabas que hagan la estimación entre ellas, quedando de la siguiente manera: *#./pathchirp –S* [ipaddress_snd] *–R* [ipaddress_rcv]

Figura 14 Pathchirp run

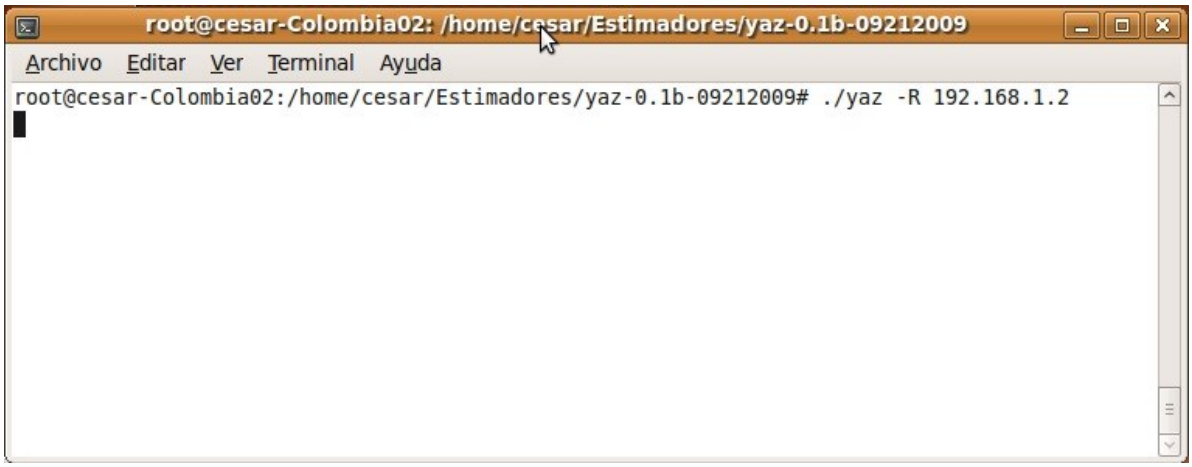
Fuente: Autor del proyecto

Yaz

- Se descomprime el archivo descargado. Previamente a la instalación hay que instalar el paquete *g++,* pues es el compilador que utiliza. Se accede a la carpeta donde quedó descomprimido a través de la consola y se ejecuta: *./configure*, consecutivamente *make*.

- En la máquina cliente se compila la herramienta de la siguiente manera: *#./yaz –R* [ipaddress].

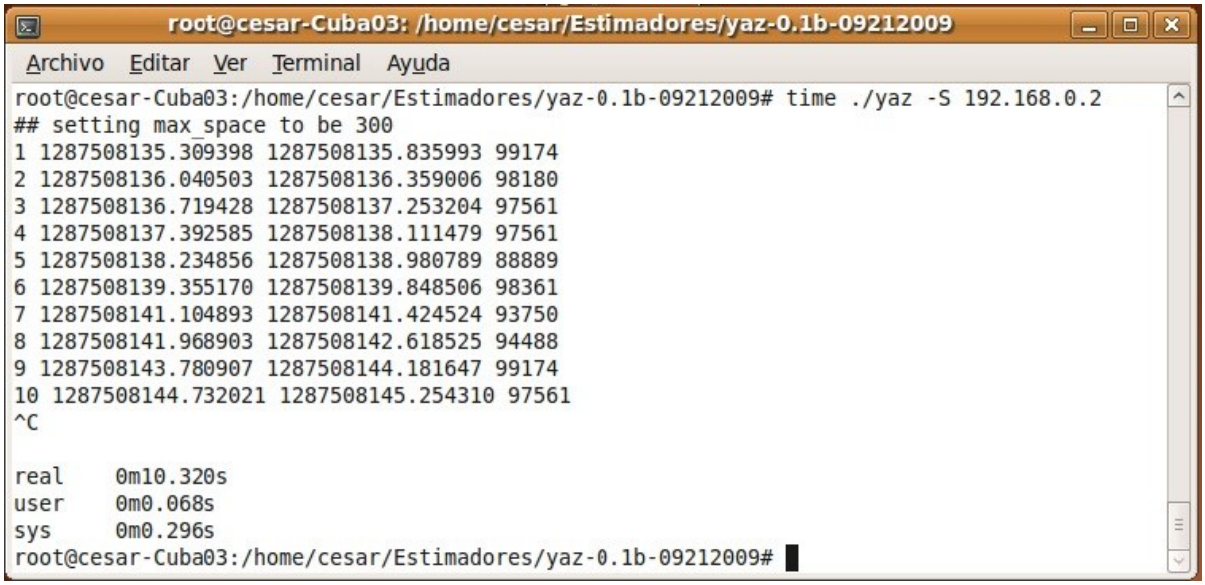
Figura 15 Yaz reciver



Fuente: Autor del proyecto

- En la máquina receptor se compila la herramienta de la siguiente manera: *#./yaz –S* [ipaddress].

Figura 16 Yaz sender



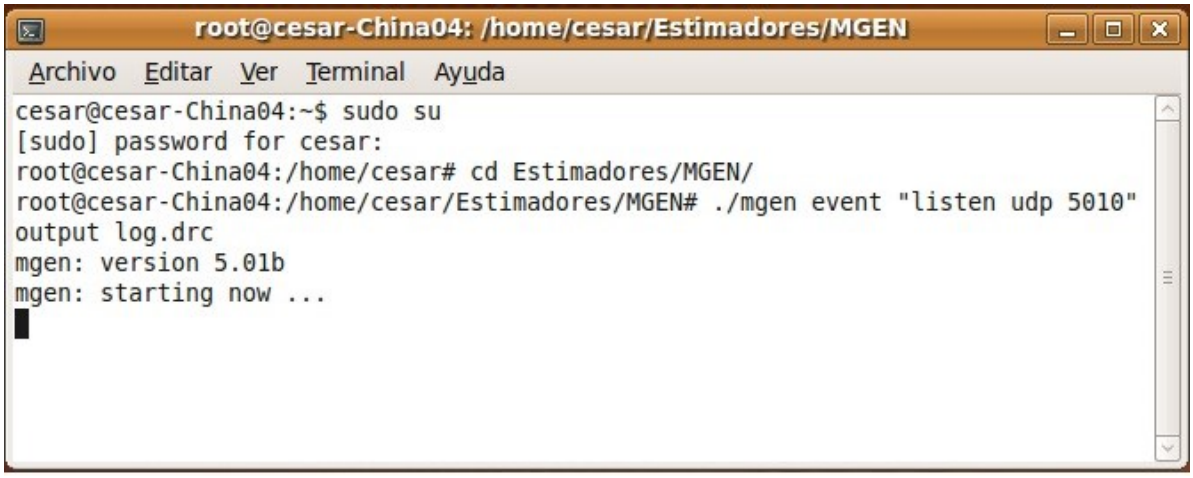
Fuente: Autor del proyecto

MGEN

- Solo se descomprime el archivo descargado.

- En la máquina receptor se compila la herramienta de la siguiente manera: *#./mgen event "listen udp 5010" output log.drc*

Figura 17 Mgen listen



Fuente: Autor del proyecto

- En la máquina cliente se compila la herramienta de la siguiente manera: *./mgen event "ON 1 UDP DST <ipaddress>/<port> <tipo de tráfico> [<# de paquetes> <cantidad de bits>]" INTERFACE eth0*

Figura 18 Mgen sender

Fuente: Autor del proyecto

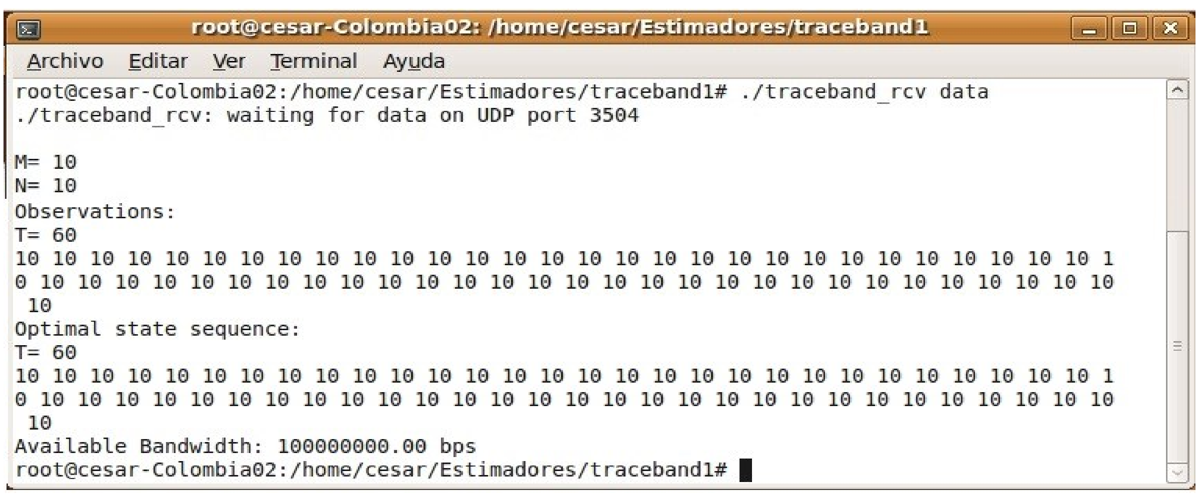
Traceband

Para el montaje de esta herramienta se le creó un *makefile*, debido a que no contaba con esta facilidad para su montaje. El comando *make* es usado en las labores de creación de ficheros ejecutables o programas para su instalación, la limpieza de los archivos temporales en la creación del fichero, todo ello especificando los parámetros iniciales que deben estar en el *makefile* al este ser ejecutado.

- Se descomprime en cada una de las máquinas cliente y receptor, se accede a la carpeta a través de la consola y se ejecuta sencillamente *make*, quedando ya en ellas los ficheros de ejecución *traceband_snd* y *traceband_rcv.*

- En la máquina receptor se compila el software de la siguiente manera: # ./traceband_rcv

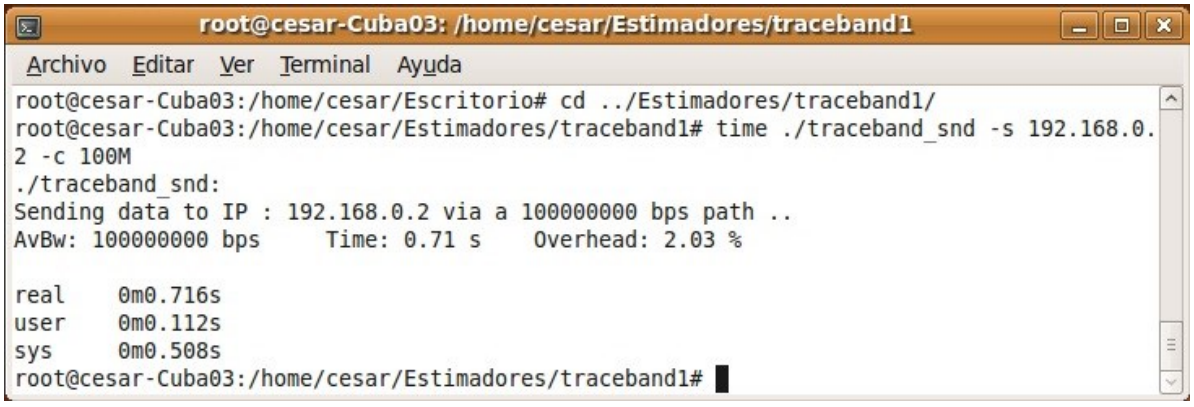
Figura 19 Traceband reciver



Fuente: Autor del proyecto

- En la máquina cliente se compila el software de la siguiente manera: # *./traceband_snd* –*s* [ipaddress] –c [Path capacity] M

Figura 20 Traceband sender

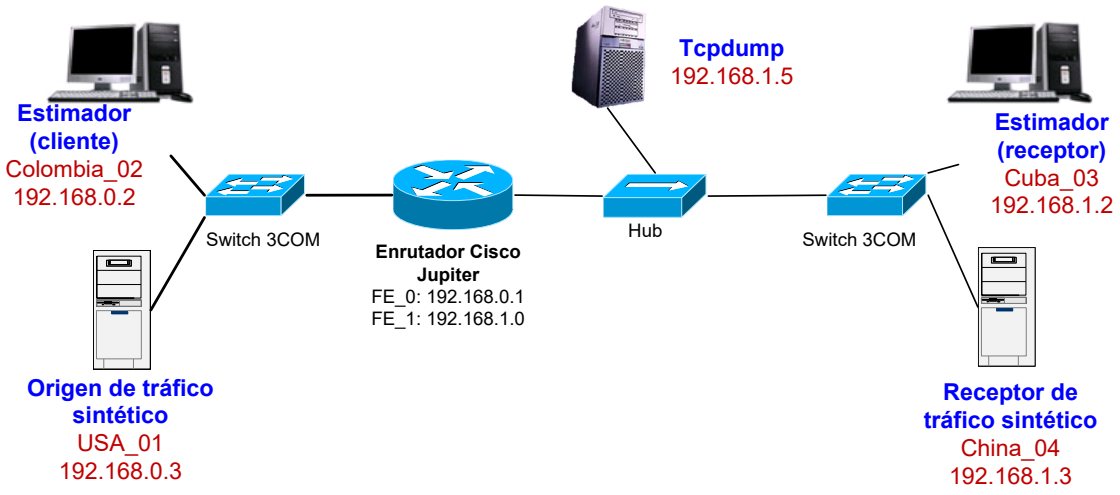


Fuente: Autor del proyecto

4.2 DESCRIPCIÓN DEL TESTBED

El *testbed* que se implementó cuenta con 5 computadores de recursos medios, procesador Pentium 4 y memoria RAM entre 512 mb y 1 Gb. Se instaló en cada una de las cinco computadoras el sistema operativo Linux, específicamente la distribución Ubuntu 9.04. Las máquinas se conectan a través de un router Cisco serie 1800 y dos switches 3 COM (ver Figura 21 Testbed de red) Las máquinas cuentan con las herramientas de estimación respectivas instaladas para hacer las evaluaciones.

Figura 21. Testbed de red.



Fuente: Autor del proyecto

Las máquinas *Estimador* (cliente) y *Estimador* (receptor) fueron utilizadas para las estimaciones, mientras que las máquinas *Origen* y *Receptor* de tráfico sintético fueron utilizadas para la generación de tráfico sintético o artificial en la red. El equipo *Tcpdump* fue el encargado en capturar las trazas para hacer los cálculos de *overhead* y *error* en las estimaciones, ya que solo *Traceband* brinda toda la información a tener en cuenta para hacer las evaluaciones.

Se estableció comunicación con el enrutador por la consola de un ordenador a través de *minicom* por el puerto serie, donde se configuraron las interfaces fast ethernet para tener comunicación entre las dos redes que se crearon, la 192.168.0.0/24 y la 192.168.1.0/24.

El *testbed* construido tiene la siguiente funcionalidad. Las máquinas con número de IP 192.168.0.2 y 192.168.1.2 se utilizan para albergar las herramientas de estimación a ser evaluadas. Por esa razón se indica una como *Cliente* y la otra como *Receptor* de los paquetes de prueba que son enviados por las herramientas. La máquina con *tcpdump* se utiliza para capturar trazas del tráfico que pasa en el

enlace inmediatamente posterior al enrutador. Este enlace va a ser el cuello de botella en la comunicación entre las maquinas de las dos diferentes redes. Lo anterior dado que el enrutador Cisco posee interfaces que operan a 1 Gbps y el hub puede soportar como máximo 100 Mbps de capacidad de enlace. Finalmente, las máquinas con números de IP 192.168.0.3 y 192.168.1.3 son las encargadas de generar y recibir tráfico sintético y por lo tanto albergan el generador de tráfico *Mgen*.

4.3 DISEÑO DE LOS EXPERIMENTOS

Para evaluar las herramientas seleccionadas se diseñaron un grupo de escenarios con diferentes tipos de tráfico, dado que la herramienta generadora de tráfico *Mgen* cuenta con esta capacidad.

De acuerdo con el tipo y la cantidad de tráfico (ver Tabla 1), se diseñaron 10 escenarios de pruebas, donde se corrieron 5 experimentos para cada herramienta. Es decir, que en total se realizaron 5 experimentos x 10 distintos escenarios x 7 herramientas de estimación. Esto implica un total de 350 experimentos.

Tabla 1. Escenarios definidos para los experimentos.

Sin tráfico			
Periódico	30%	50%	70%
Poisson	30%	50%	70%
Burst	30%	50%	70%

El tráfico *Periódico* consiste en el envío constante de paquetes por segundo.

El tráfico *Poisson* corresponde a una mejor aproximación a lo que es el tráfico que se observa en un sistema de comunicaciones. En este los tiempos entre paquetes se distribuyen de manera exponencial.

El tráfico *Burst* consiste en el envío de paquetes por ráfagas; en cada ráfaga los tiempos entre paquetes son en forma exponencial.

4.4 MÉTRICAS UTILIZADAS

Para evaluar la operación de las herramientas, se tuvieron en cuenta tres métricas *tiempo de estimación*, *overhead* y *error de estimación*.

4.4.1. Tiempo de estimación. Está definido como el tiempo medido en segundos, que se toma la herramienta desde que es llamada hasta que se arroja un resultado.

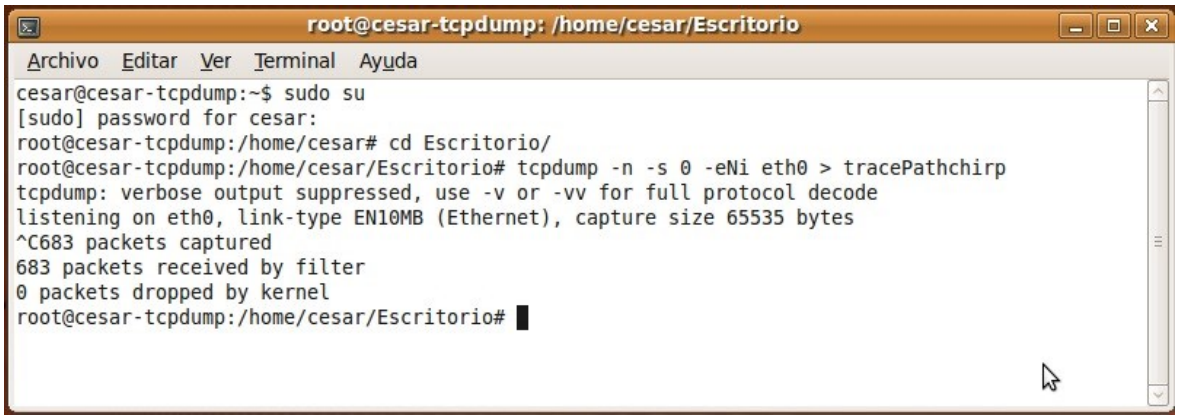
4.4.2. Overhead. Es la cantidad de bits por unidad de tiempo que debe insertar el estimador para hacer la estimación. Se mide en bits por segundo (bps).

4.4.3. Error de estimación. Se define como la relación entre el valor estimado y el valor real y se mide en términos porcentuales: (Ancho de banda real - Ancho de banda estimado)/Ancho de banda real.

4.5 PROCESAMIENTO DE LA INFORMACIÓN

Para el procesamiento de la información se hizo la captura del tráfico en cada estimación entre las máquinas a través de *tcpdump* en el Equipo *Tcpdump* (ver Figura 22) para hacer los cálculos de *overhead* y *error de estimación* de las herramientas.

Figura 22 Tcpdump



Fuente: Autor del proyecto

Para el procesamiento de esta información se hizo uso de un script *GetTraffic.awk* (ver Figura 23 y 24 Código del script GetTraffic). el cuál calcula el tráfico total real que estaba circulando en el momento en la red entre las máquinas y el tráfico generado entre los equipos que hicieron las estimaciones. Esta información se recogió en cada una de las trazas a través de *tcpdump*; obteniéndose así los verdaderos resultados de ancho de banda ocupado en cada estimación, para después comprobar con la estimación dada por las herramientas y así ver cuán cercano estaba el resultado estimado al real. Los comandos son mostrados en la Figura 22 Tcpdump.

Figura 23 Código del script GetTraffic (1)

```
#!/usr/bin/gawk -f
# Reads a tcpdump trace generated by:
#   tcpdump -s 0 -eNi <interface name>
# Example: 23:35:16.407441 00:64:40:31:a2:3b > 00:11:25:0f:25:48, ethertype IPv4 (0x0800),
# length 1514: 192.168.0.3.42871 > 192.168.1.3.5010: UDP, length 1500

# And generates this trace:
#   <time in seconds> <ethernet frame length> <0:tool traffic 1:cross traffic> <bandwidth bps>

BEGIN {num_bytes = 0
       tool_bytes = 0
       tool_pckts = 0
       flag = 0          # To vertify if first tool packet has been found
       cross_traffic = 0
       FS="[ ]"}         # FS is the field separator
{
  if ($1 != "") {       # Verify that the trace is not empty
    # get the timestamp and split it into hour, minute and second
    split($1,a,":");    # split hours and minutes
    split(a[3],b,".");  # split second into second and microsecond
    split($9,c,":");    # the packet size will be stored in c[1]
    if ($8!="length")   # Then this is a wird packet (it will not be counted)
      c[1] = 0;
    split($10,d,".");
    split($12,f,".");
    cross_traffic = ( ((d[3]=="1" && d[4]=="2") || (f[3]=="1" && f[4]=="2"))?0:1);
    if (NR == 1) {
      start_time[1] = a[1];
      start_time[2] = a[2];
      start_time[3] = b[1];
      start_time[4] = b[2];
      printf "%4.8f %i %i %10.2f\n",0.0,c[1],cross_traffic,0.0 > "data.txt";
    }
    else if (c[1] >0) {
      curtime = ((a[1]-start_time[1])*3600 + (a[2]-start_time[2])*60 + (b[1]-start_time[3]))
                *1000000 + (b[2]-start_time[4]);
      if (cross_traffic == 0) {
        if (flag == 0) {
          flag = 1;
          first_tool[1] = a[1];
          first_tool[2] = a[2];
          first_tool[3] = b[1];
```

Fuente: Autor del proyecto

Figura 24 Código del script GetTraffic (2)

```
          first_tool[4] = b[2];
        }
        else {
          tooltime = ((a[1]-first_tool[1])*3600 + (a[2]-first_tool[2])*60 + (b[1]-first_tool[3]))
                     *1000000 + (b[2]-first_tool[4]);
        }
        tool_bytes += c[1];
        tool_pckts ++;
      }
      printf "%4.8f %i %i %10.2f\n",curtime/1000000.0, c[1],cross_traffic,c[1]*8/
             ((curtime-prevtime)/1000000.0) >> "data.txt";
      prevtime=curtime;
    }
    num_bytes += c[1];
  }
}
END {print "number of packets, number of bits, time taken (s), and bandwidth (bps)"
     print "-------------------------------------------------------------------------"
     print "General traffic captured:"
     total_bw = (num_bytes*8)/(curtime==0?0.000001:curtime/1000000.0)
     printf "%d\t %d\t %10.2f\t %10.2f\n", NR, num_bytes*8, curtime/1000000.0, total_bw
     print "Estimation tool traffic captured:"
     tool_bw = (tool_bytes*8)/(tooltime==0?0.000001:tooltime/1000000.0)
     printf "%d\t %d\t %10.2f\t %10.2f\n", tool_pckts, tool_bytes*8, tooltime/1000000.0, tool_bw
     printf "Tool overhead: %5.2f\n", tool_bw*100/100000000
    }
```

Fuente: Autor del proyecto

El tiempo de estimación se calculó con una función, ésta da el tiempo que un proceso se demora ejecutando en la consola, esto se hizo para cada estimación en la máquina *Estimador* (cliente) (ver Figura 20 Traceband sender).

Podemos observar en este caso que la herramienta *Traceband* nos brinda la información del tiempo que se demoró en hacer la estimación y si comparamos este con el dado por la función *time* difiere de 0.01s, por lo que es un resultado muy cercano al real. Este tiempo se midió en cada experimento hecho con las herramientas que no daban este resultado (*Yaz, Pathchirp, Spruce y Igi-Ptr*).

5. RESULTADOS

Para hacer el análisis de los experimentos se contó con una función en *Matlab*, *error_bytool.m*, la cual procesa la información de cada herramienta y la obtenida en la captura de las trazas desde la máquina con *tcpdump*. A esta información recogida en un documento se le aplicó una *t-Student* implementada en la función *error_bytool.m* (ver Figura 25 Error by tool) para obtener un intervalo de confianza, obteniéndose para cada prueba las gráficas que más adelante se mostrarán.

Figura 25. Error by tool

```
clear; clc;
close all;
alpha = 0.05;
error=load('data/error_sin_trafico.txt');
tools = {'Pathload'; 'Spruce'; 'Traceband'; 'Yaz'; 'IGI';
'Pathchirp'; 'PTR'};

set(0,'DefaultAxesFontSize',12);
orient portrait;
for k=1:length(tools)
    avg(k)=mean(error(k,1:5));
    [h,p,ci]=ttest(error(k,1:5),avg(k),alpha);
    dev(k)=ci(2)-avg(k);
end

Q=errorbar(avg*100,dev*100,'xr');
set(Q, 'LineWidth', 2);
set(Q, 'MarkerSize', 10);
set(Q, 'Color', 'b');

set(gca,'XTick',[1:7])
set(gca,'XTickLabel',tools)
set(gca,'YGrid','on')
xlabel('{\bf Herramienta de estimación}', 'FontSize',16);
ylabel('{\bf Error de Estimación (%)}', 'FontSize',16);

print -djpeg 'figuras/error_sin_trafico.jpg' -r600;
```

Fuente: Autor del proyecto

Aclarar al lector que cuando se hable en este documento de intervalos de confianza grande o pequeños se estará refiriendo a la dispersión de los resultados arrojados por las herramientas en los mismos tipos de pruebas.

Cabe destacar que en la herramienta *Yaz* y *Pathchirp* los *tiempos de estimación* y *overhead* se calcularon haciendo una división de entre la cantidad de estimaciones realizadas para así tener un valor puntual de *overhead* y *tiempo* en cada estimación, debido a que éstas herramientas no cuentan con forma de hacer estimaciones individuales.

Obsérvese también que de la herramienta *Igi-Ptr* se dan dos resultados por separados, *Igi y Ptr*, pues aunque son dos técnicas diferentes que aplican para hacer la estimación éstos compartieron los mismos resultados, tanto en *overhead* como *tiempo de estimación*.

Figura 26. Error de estimación sin tráfico

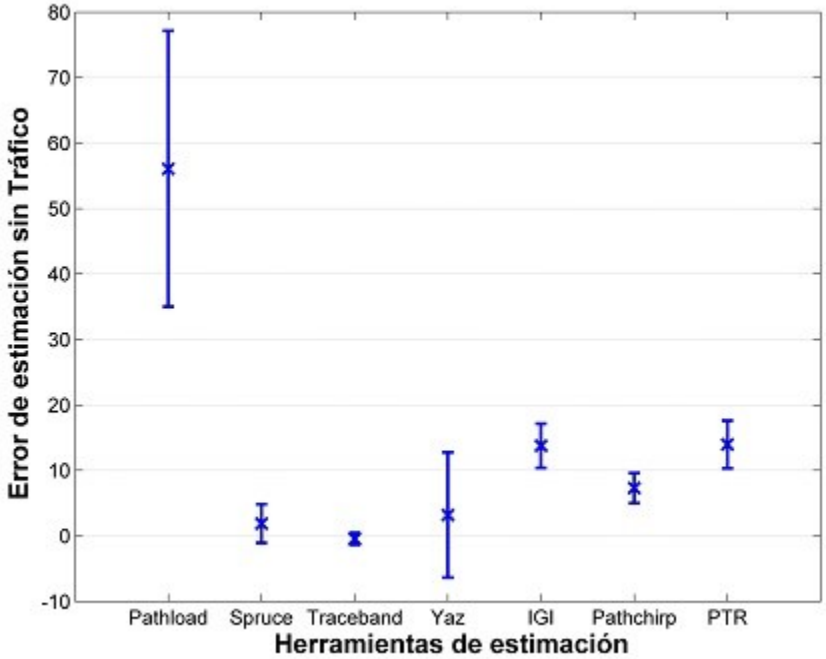


Figura 27. Overhead en pruebas sin tráfico

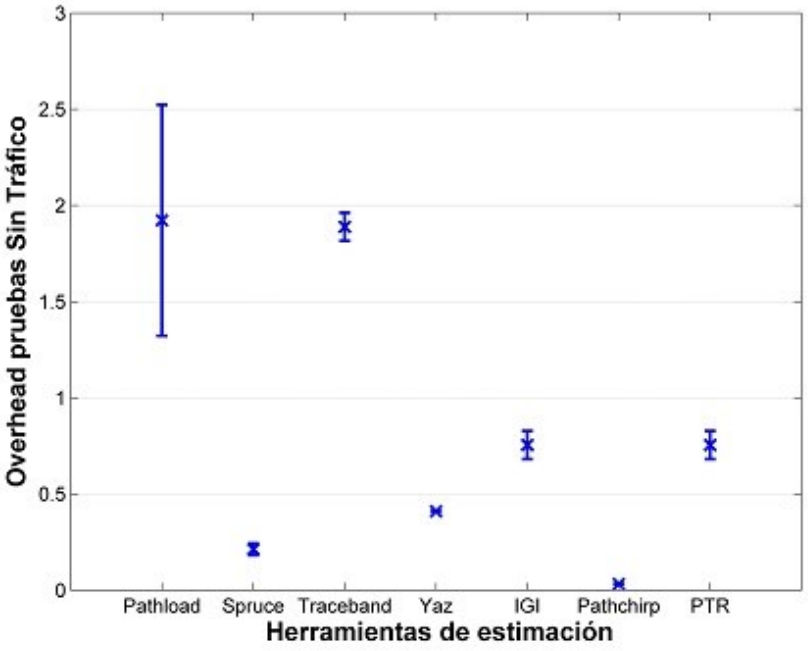
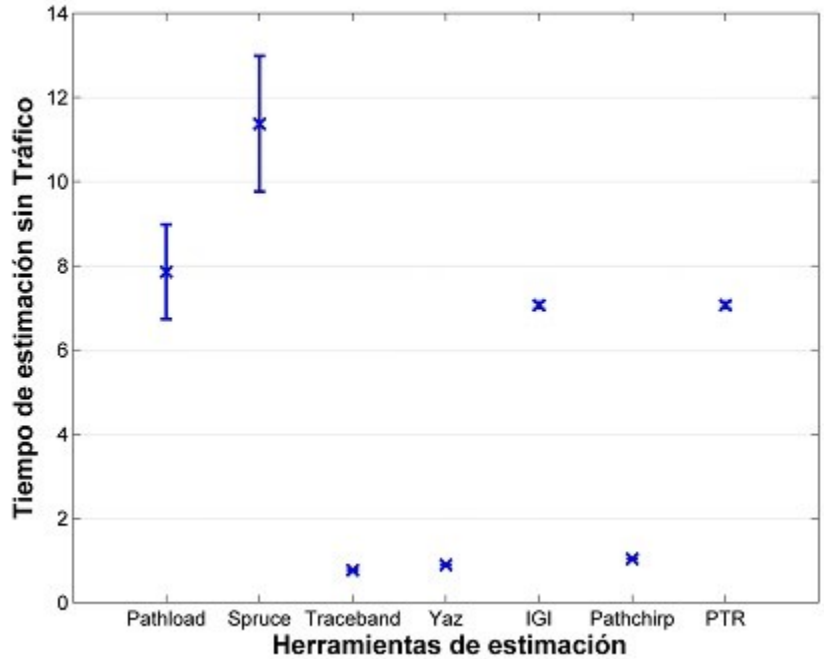


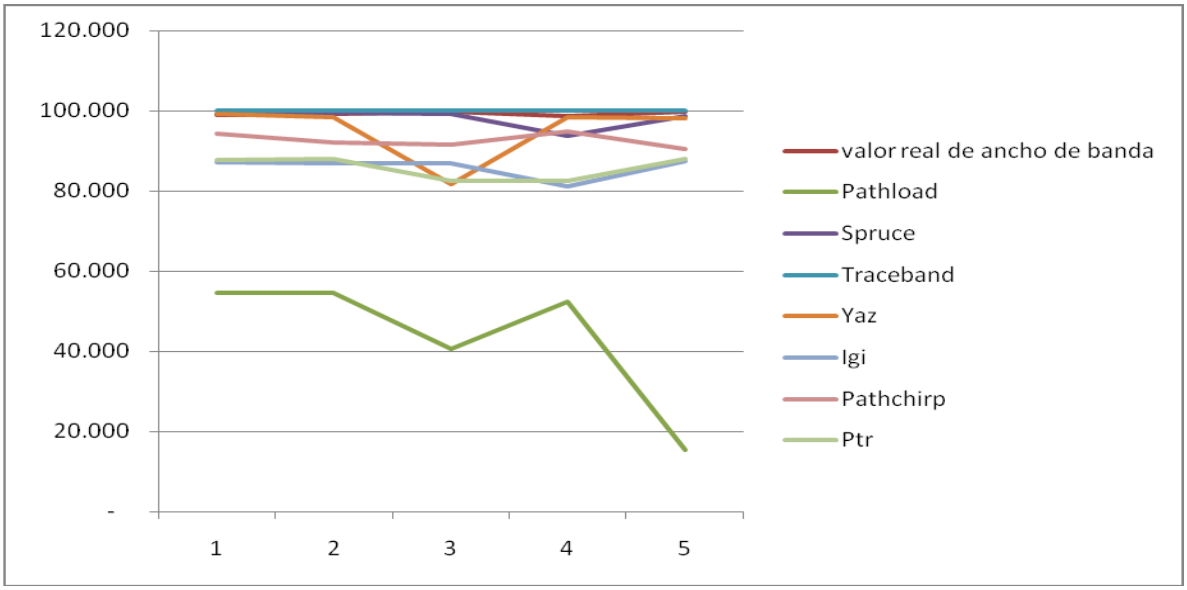
Figura 28. Tiempo de estimación sin tráfico



A pesar que se analizaron los resultados de *tiempo de estimación* y de *overhead* para cada experimento realizado, no se mostrarán más las gráficas obtenidas para estos indicadores, pues al no haber una variación notable en éstos se dejará solo la notación que permanecen muy similares en todos los experimentos y se procederá a hacer el análisis en los *errores de estimación* que sí varían.

Error de estimación: En los experimentos hechos sin tráfico se puede observar como muestra la Figura 8, que la herramienta más precisa es *Traceband*, dado que su intervalo de confianza es bien pequeño con una media sobre el 0% de *error*. En *Yaz y Spruce* también se obtuvieron buenos resultados, sus intervalos de confianzas pasan sobre el valor real de ancho de banda disponible. Las demás herramientas tienen un comportamiento irregular y sus intervalos de confianzas no están cercanos al valor cero; *Pathload* es la herramienta más imprecisa para este tipo de pruebas.

Figura 29 Estimaciones de las herramientas para pruebas sin tráfico



Overhead: En este tipo de experimento como en los demás por las razones antes explicadas las herramientas *Traceband* y *Pathload* son las de mayor *overhead*, por lo que son las que más paquetes tienen que insertar para hacer las estimaciones.

Tiempo: Las herramientas que más demoran en hacer las estimaciones en todos los experimentos son *Spruce* y *Pathload*, con una media de 11 y 8 segundos respectivamente. La herramienta *Traceband, Yaz y Pathchirp* son las que menos tiempo toman en hacer las estimaciones.

Figura 30. Error de estimación Periódico al 30%

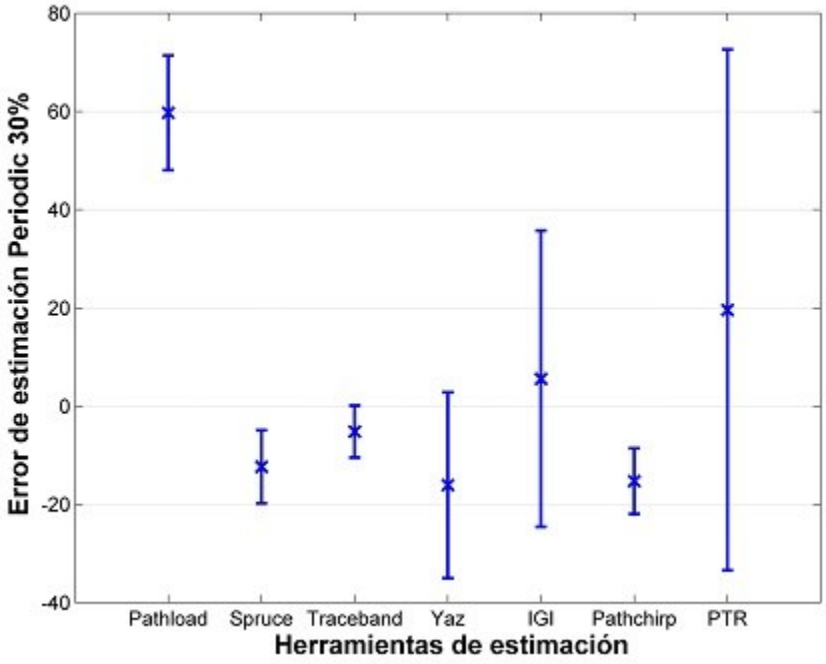
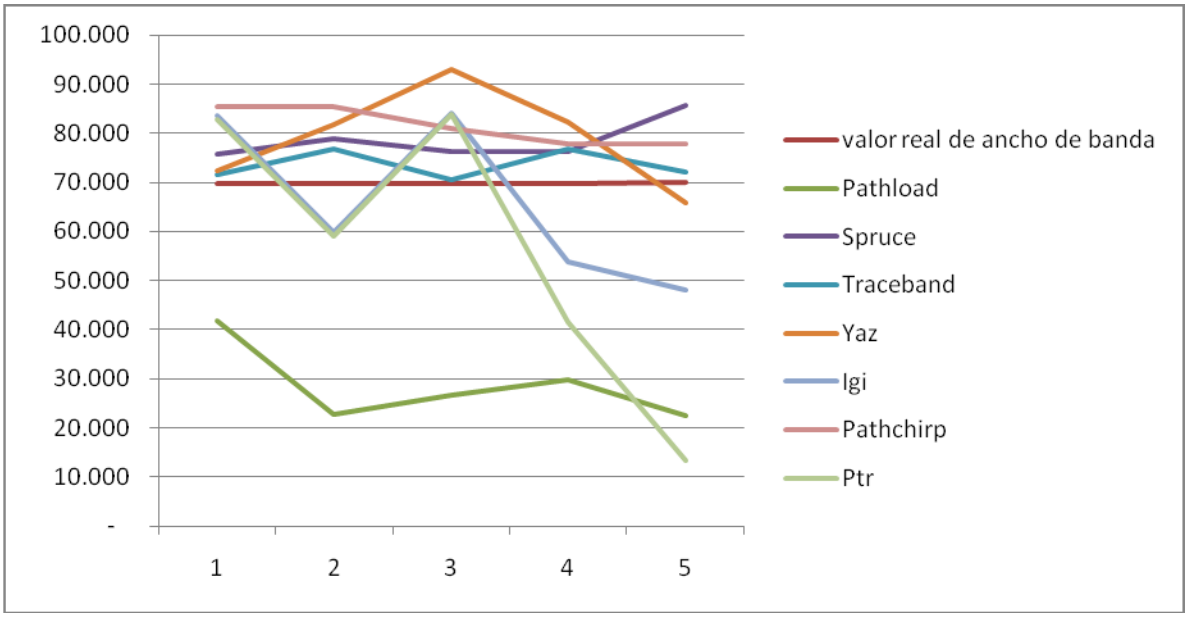


Figura 31 Estimaciones con tráfico Periódico al 30%



Para este tipo de pruebas se obtuvo que *Traceband* es la herramienta con mejores resultados arrojados, obteniéndose en ella una media muy cercana a cero (0%) y con un intervalo de confianza bien pequeño, por lo que la mayoría de estimaciones que hace están muy cercanas al valor real.

En *Spruce y Pathchirp* también se obtuvieron resultados similares aunque un poco más alejado del real. *Igi y Ptr* fueron las herramientas que más inestabilidad tuvieron a la hora de hacer las estimaciones, debido a que su intervalo de confianza es bien grande.

Figura 32. Error de estimación Periódico al 50%

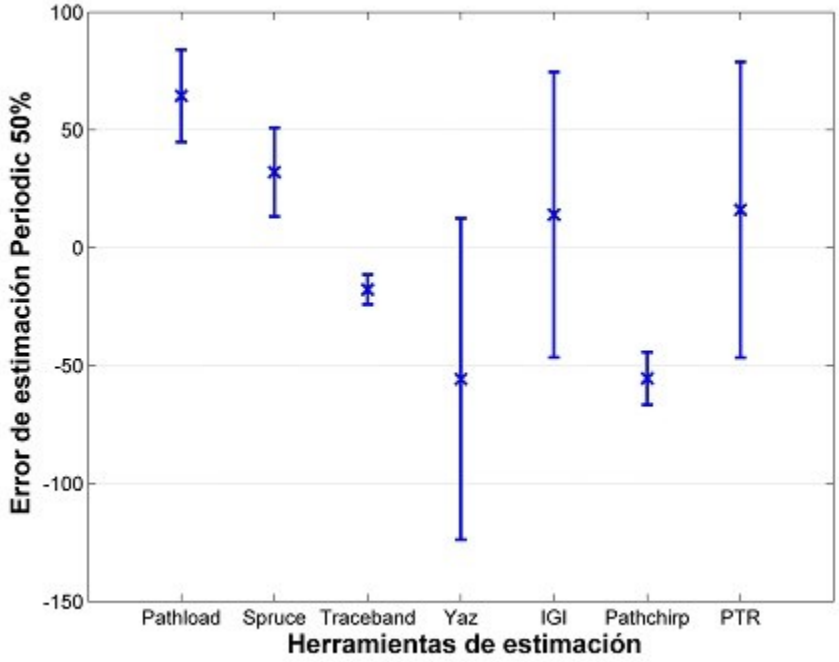
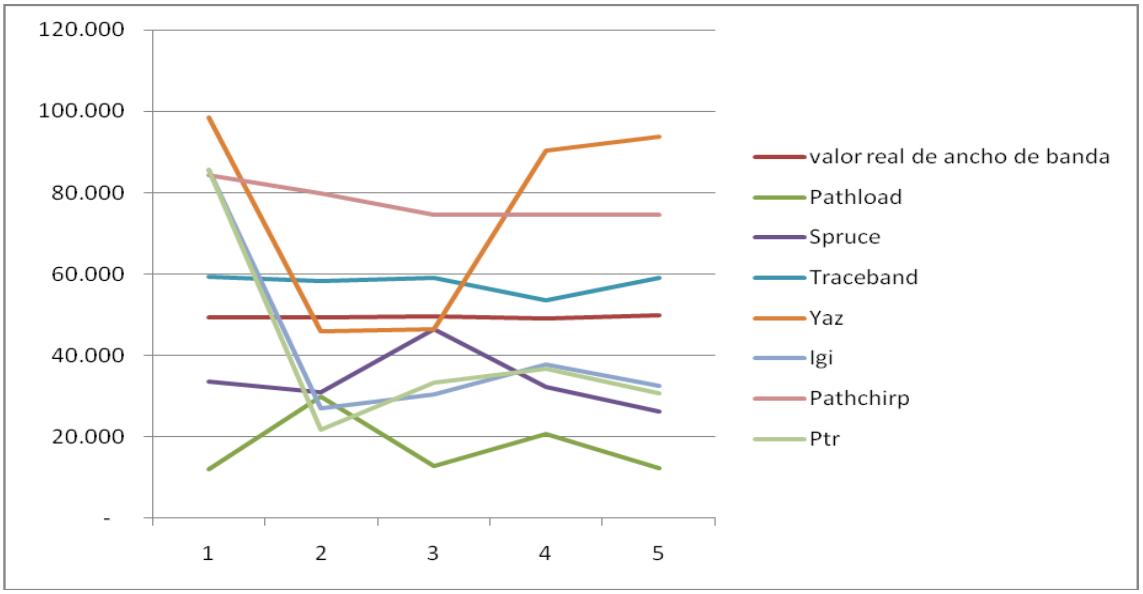


Figura 33. Estimaciones con tráfico Periódico al 50%



En este tipo de experimento continúa la herramienta *Traceband* siendo la de mejores resultados, pues su media está bien cercana al valor real y su intervalo de confianza es bien pequeño.

Las herramientas *Igi y Ptr* aunque tienen una media muy cercana al valor real su intervalo de confianza es bastante grande, debido a su inestabilidad a la hora de hacer las estimaciones puntuales.

Pathload continúa dando resultados alejados del real aunque sus estimaciones se van acercando más al valor real.

Figura 34. Error de estimación Periódico al 70%

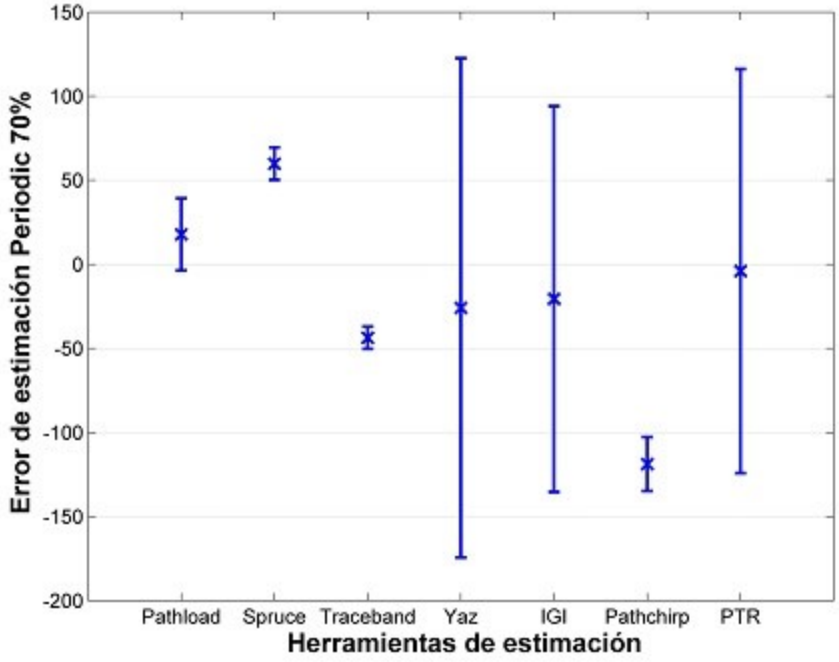
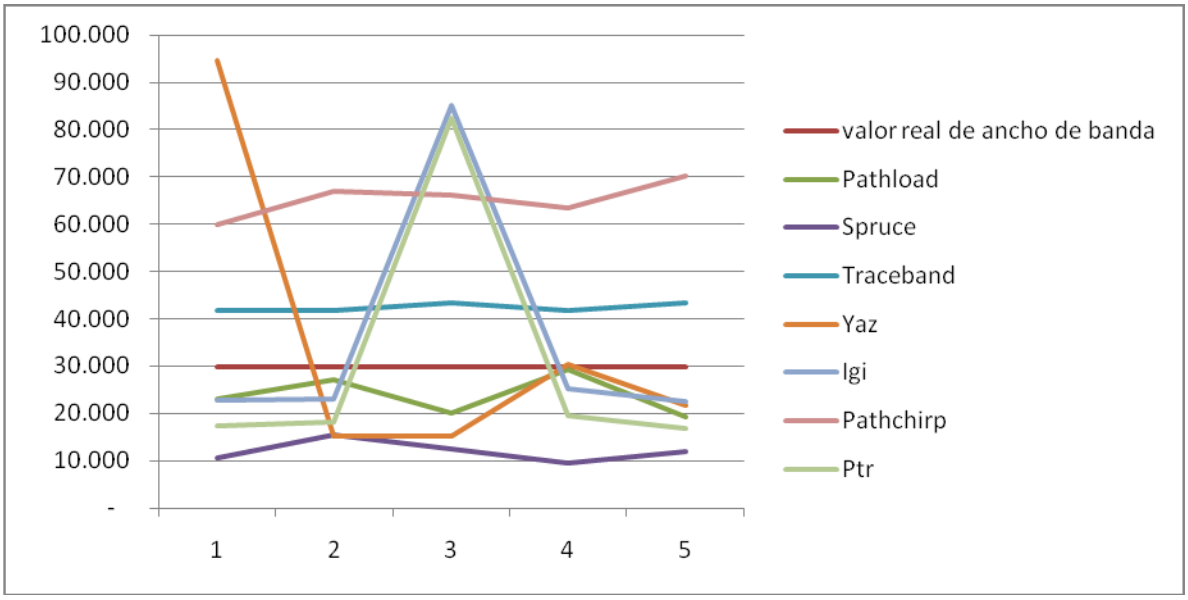


Figura 35. Estimaciones con tráfico Periódico al 70%



Para este tipo de experimentos, *Pathload* es la herramienta con mejor comportamiento, debido a que su intervalo de confianza es pequeño y pasa por el valor real.

En *Traceband* se obtuvieron resultados ya más alejados del valor real en comparación los experimentos anteriores. Muy similar pero con diferente sentido se encuentran los resultados obtenidos con *Spruce*.

Yaz es la herramienta con más inestabilidad para este tipo de experimentos, esto se puede ver en el amplio intervalo de confianza en el que se encuentran sus resultados.

Figura 36. Error de estimación Poisson al 30%

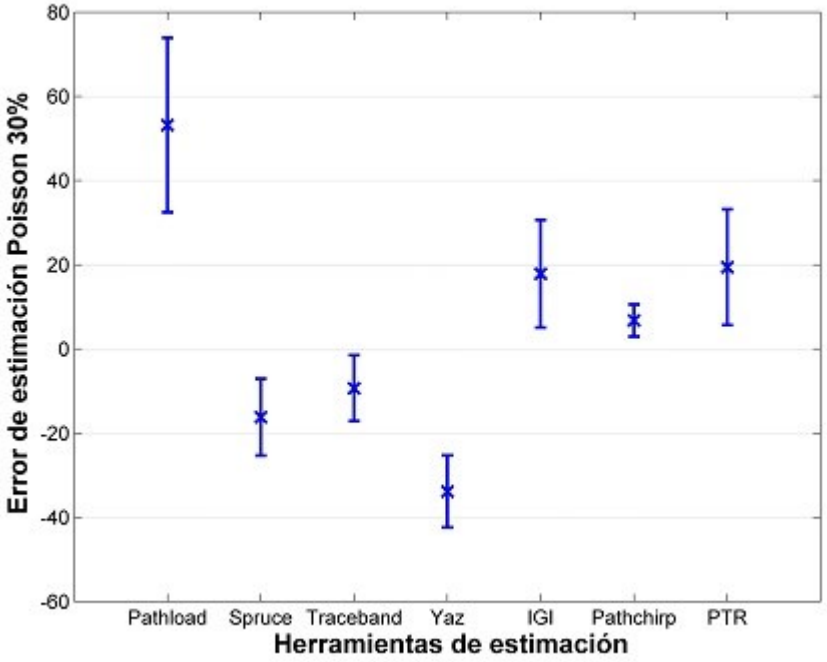
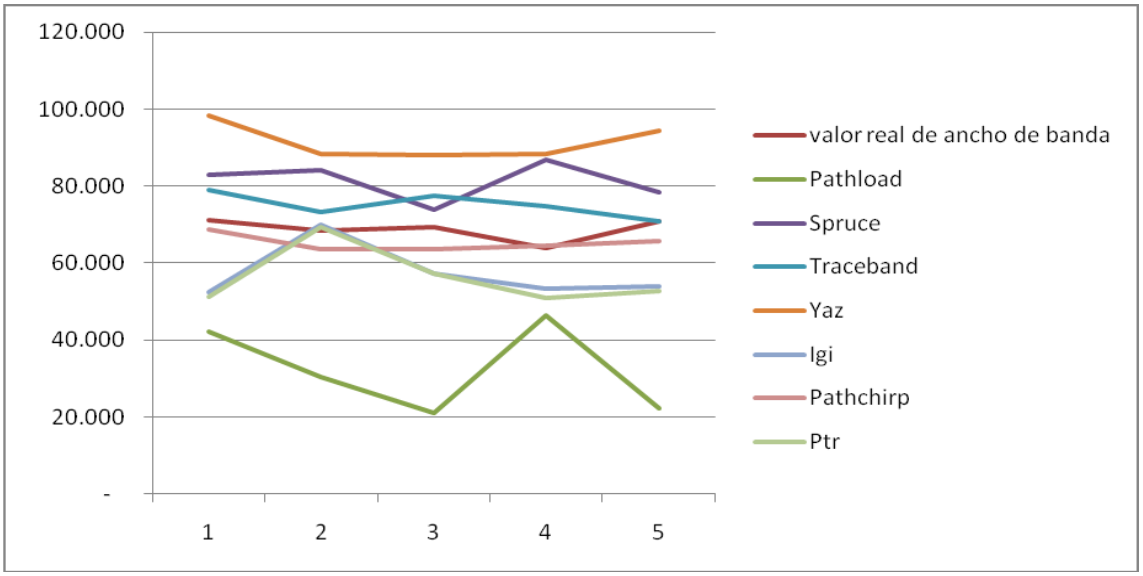


Figura 37. Estimaciones con tráfico Poisson al 30%



En este tipo de experimentos la herramienta que más se ajusta al valor real es *Pathchirp*, debido a que su media está muy cercana al valor real con un intervalo de confianza bien pequeño.

En *Traceband* se obtuvieron buenos resultados debido a que su intervalo de confianza sigue siendo bien pequeño y se acerca mucho al valor real. *Spruce* tiene un comportamiento muy similar con respecto al intervalo de confianza pero su media está un poco más por debajo que la de *Traceband*.

De nuevo *Pathload* al haber poca congestión en el canal sigue siendo la herramienta con resultados más alejados al real.

Figura 38. Error de estimación Poisson al 50%

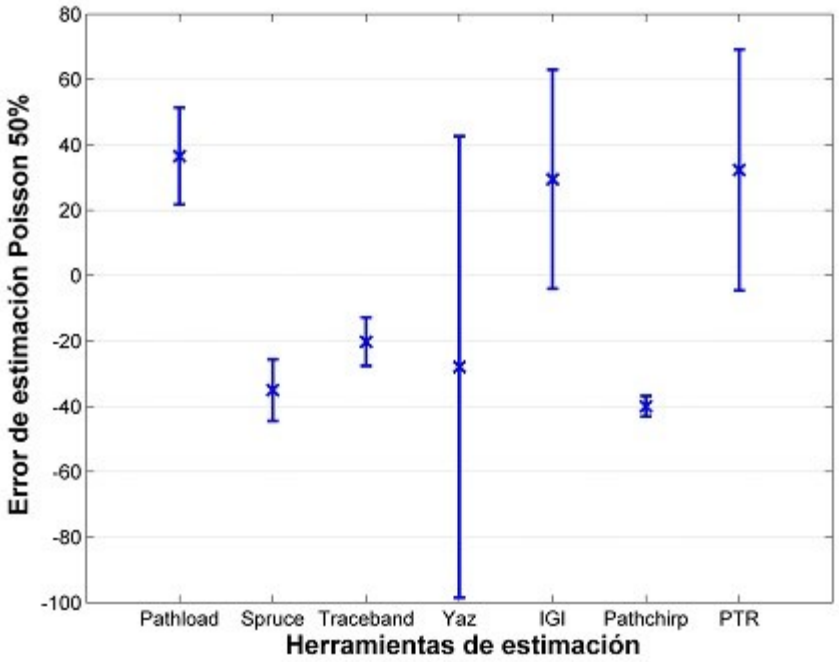
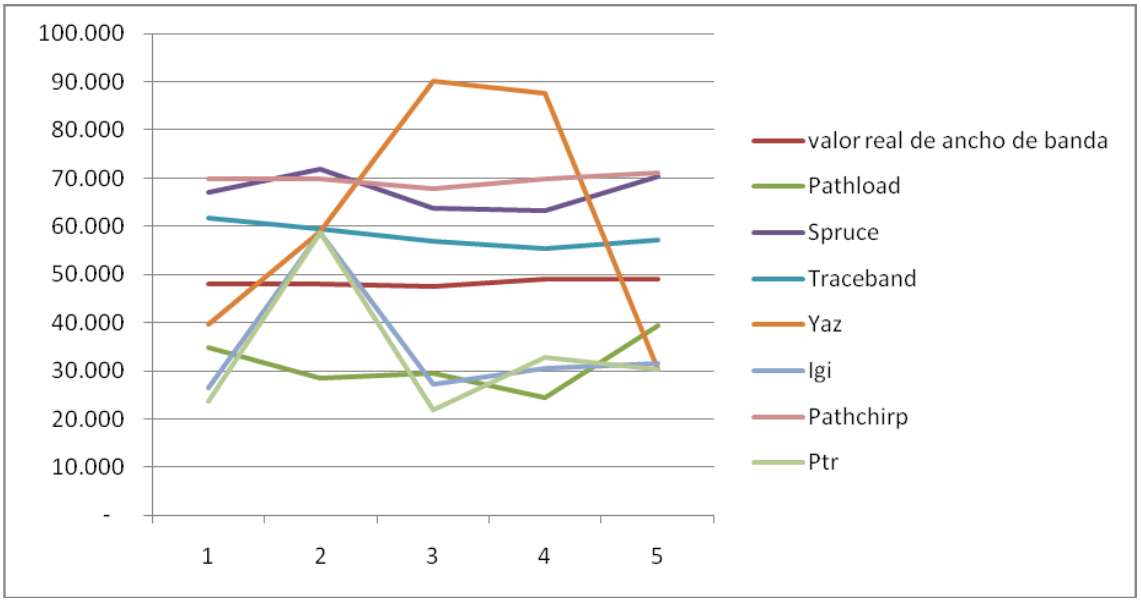


Figura 39. Estimaciones con tráfico Poisson al 50%



En este tipo de experimentos la herramienta con mejores resultados es *Traceband*, pues aunque su intervalo de confianza no pasa por el 0 que sería el valor de estimación real su media es la que más se acerca a la realidad.

Como se puede observar *Yaz* es la herramienta con más inestabilidad en este tipo de pruebas.

Pathload continúa dando resultados muy alejados, pero se va acercando un poco más al aumentar la congestión en el canal.

Spruce continúa teniendo un comportamiento similar a *Traceband* en comparación con el intervalo de confianza, pero con una media de 10 *mbps* más alejada.

Figura 40. Error de estimación Poisson al 70%

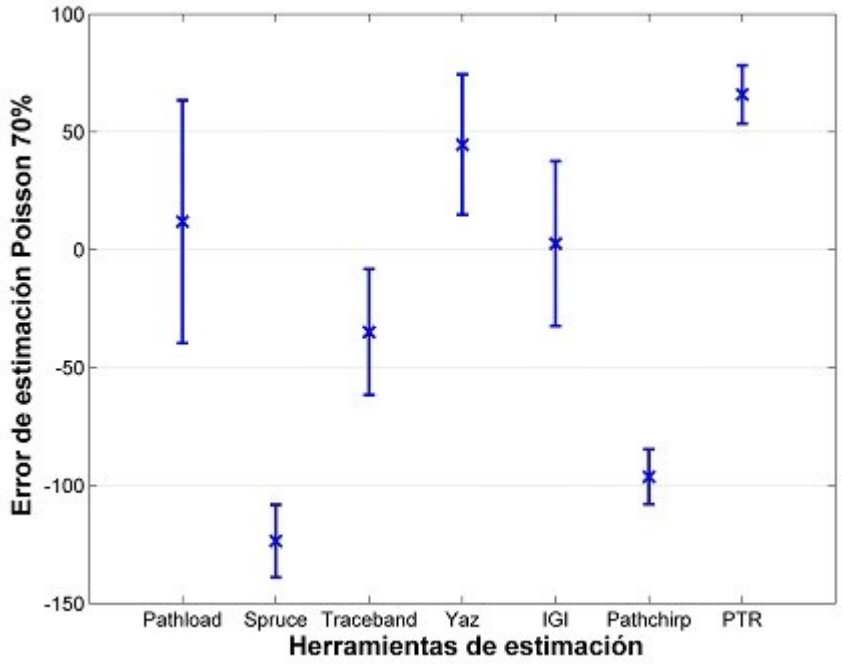
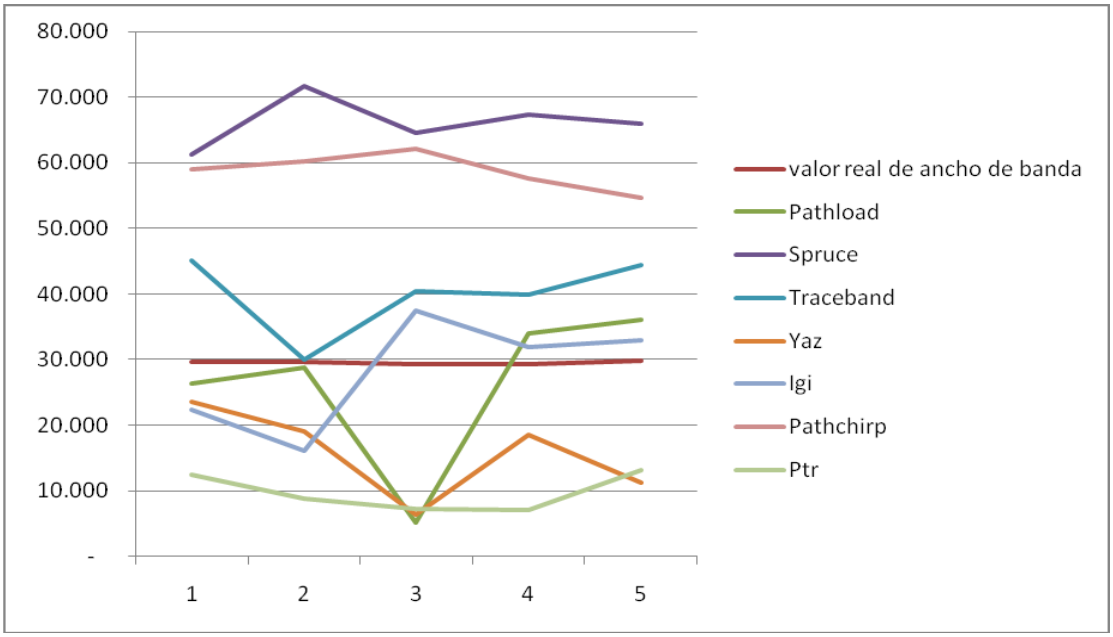


Figura 41. Estimaciones con tráfico Poisson al 70%



En este tipo de experimentos la herramienta con mejor comportamiento es *Igi* debido a que su media está muy cercana al valor real aunque con un intervalo de confianza un poco grande.

La herramienta más errática en este tipo de experimentos es *Spruce*.

En *Pathload* se obtuvieron resultados con una media muy cercana al valor real de ancho de banda disponible, pero su intervalo de confianza es bien grande por lo que tiene una gran inestabilidad a la hora de hacer las estimaciones puntuales.

Figura 42. Error de estimación Burst al 30%

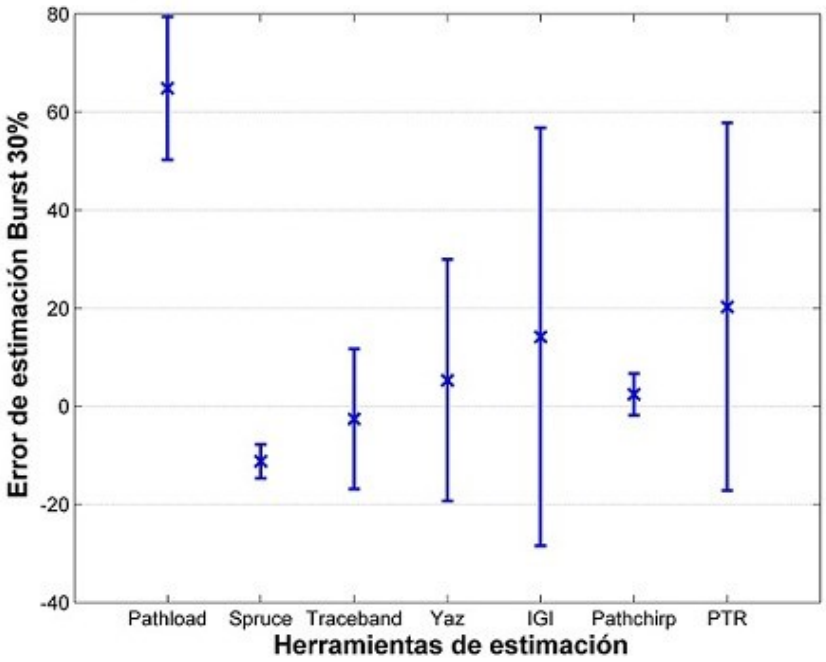
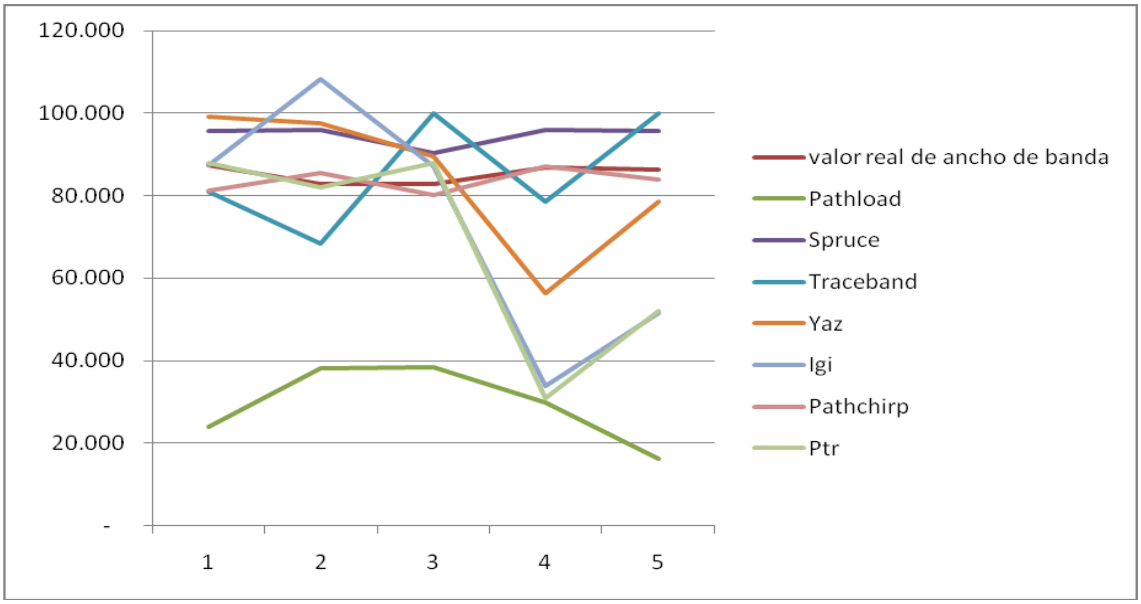


Figura 43. Estimaciones con tráfico Burst al 30%



Para este tipo de experimentos la herramienta con mejores resultados fue *Pathchirp*, ya que su intervalo de confianza es bien pequeño y pasa por el valor real de ancho de banda disponible.

Traceband cuenta con una media muy cercana al valor real de ancho de banda disponible pero su intervalo de confianza es mayor al de *Pathchirp*.

En *Spruce* se obtuvieron resultados bastante cercanos y con un intervalo de confianza bastante pequeño.

De nuevo se puede observar que *Pathload* es la herramienta más errática cuando hay poca congestión.

Figura 44. Error de estimación Burst al 50%

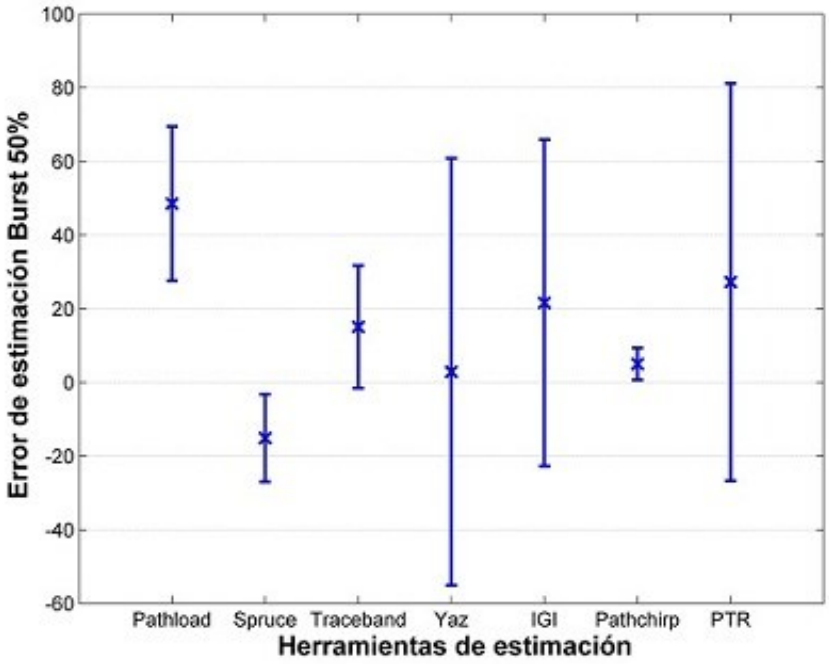
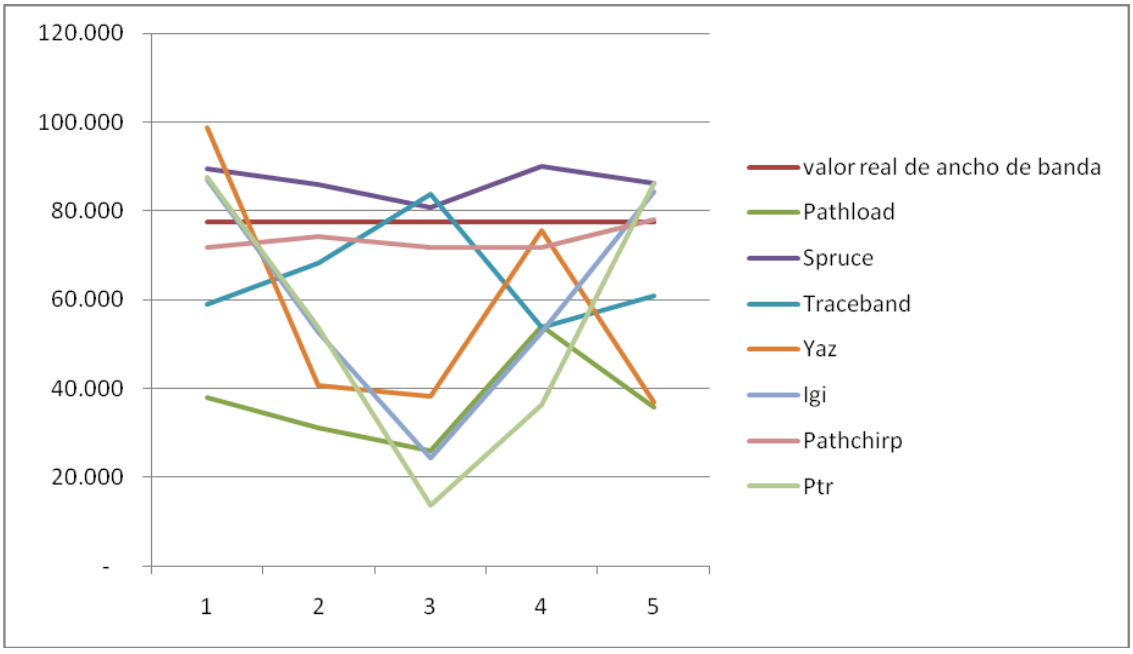


Figura 45. Estimaciones con tráfico Burst al 50%



Para este tipo de experimentos la herramienta con mejores resultados fue *Pathchirp*, ya que intervalo de confianza es bien pequeño y llega a pasar por el valor real de ancho de banda disponible.

En *Traceband y Spruce* de nuevo se puede observar un comportamiento bastante similar aunque en sentido contrario. El intervalo de confianza de *Traceband* no es tan grande y llega a pasar por el valor real de ancho de banda disponible.

Las demás herramientas tienen un intervalo de confianza bastante grande y su media está muy alejada del valor real de ancho de banda disponible.

Figura 46. Error de estimación Burst al 70%

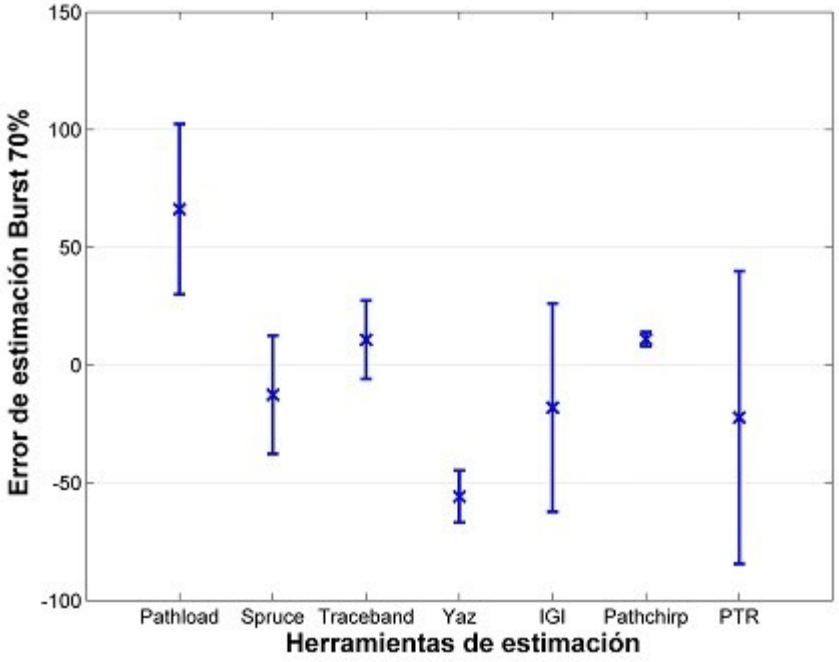
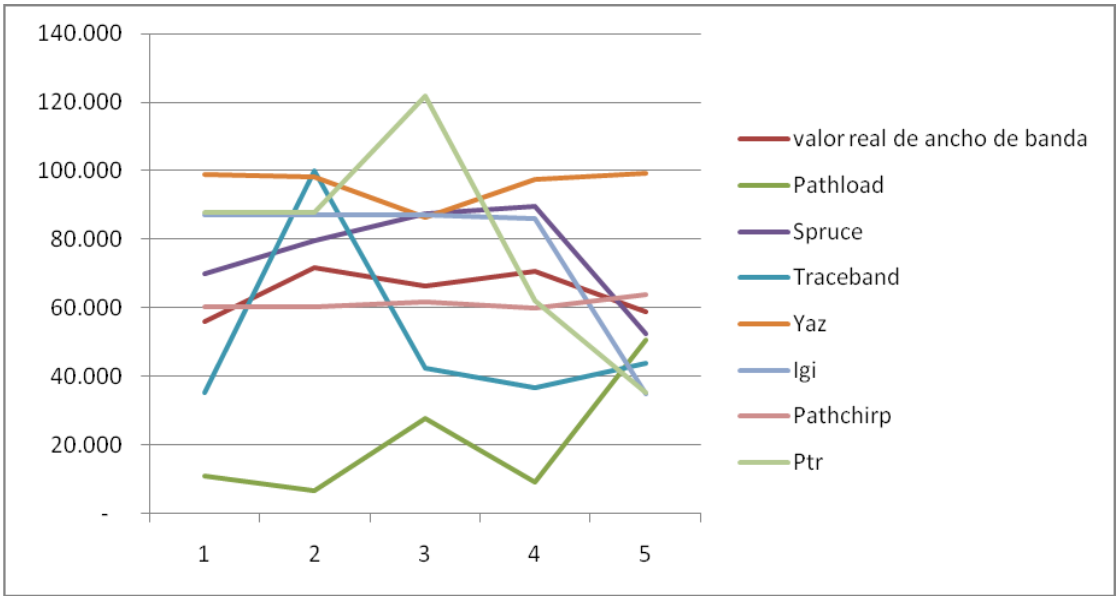


Figura 47. Estimaciones con tráfico Burst al 70%



Para este tipo de experimentos la herramienta con mejores resultados fue *Pathchirp*, ya que su intervalo de confianza es bien pequeño y su media se acerca mucho al valor real de ancho de banda disponible.

En *Traceband* se observa un comportamiento bastante cercano al valor real, su media es casi la misma que *Pathchirp* pero con un intervalo de confianza mayor al de esta.

En *Pathload, Igi y Ptr* se puede observar que sus intervalos de confianza son muy grandes por lo que muestran una baja estabilidad en su comportamiento.

6. CONCLUSIONES

En esta sección se hará un resumen individual de los resultados obtenidos por cada herramienta, concluyendo cuales herramientas tuvieron mejor comportamiento en general para los distintos escenarios definidos.

En este trabajo se contó con máquinas de recursos medios para hacer las evaluaciones de las herramientas, donde el resultado de las mismas pudo haberse alterado debido a que los cálculos requieren de gran poder computacional. En trabajos futuros se harán las mismas evaluaciones pero ya con máquinas que cuenten con mejores recursos, por lo que se pide que se tenga en cuenta cualquier conclusión a la que se llegue en este documento a la hora de hacer estos análisis, debido a las circunstancias en las que se hicieron.

En dependencia de los entornos de red donde se quiera aplicar dichas evaluaciones, se deberá tener en cuenta a través de los resultados mostrados en este estudio cuales son las herramientas más indicadas según sea el caso.

6.1 IGI

Igi es una herramienta que cuenta en la mayoría de resultados con intervalos de confianza grandes debido a su inestabilidad. Aunque la media de su intervalo de confianza a veces pasa por el valor real de ancho de banda disponible, como se pudo ver en las Figuras *Error de estimación* para los diferentes escenarios, esto se debe a la gran inestabilidad entre sus estimaciones puntuales. Solamente en

una prueba (ver Figura 16) quedó como la mejor herramienta, aunque con un *intervalo de confianza* bien grande, pero para este tipo de experimento no hubo herramienta con *media* ni *intervalo de confianza* mejor que ella.

Su *overhead* y *tiempo de estimación* permanecen muy similares en todas las pruebas con una *media* de 0.8 y 7 respectivamente.

6.2 PTR

Ptr es una herramienta que cuenta en la mayoría de resultados con un intervalo de confianza bien grande debido a su inestabilidad. Tiene un comportamiento muy similar a *Igi* en algunas pruebas, aunque no quedó de mejor herramienta para ningún tipo de experimento. Es una herramienta con un margen de error muy alto, superior al 60 % en muchos de los experimentos y gran inestabilidad como se pudieron observar en las gráficas de *Estimaciones* con los diferentes tráficos.

Su *overhead* y *tiempo de estimación* permanecen constantes en todas las pruebas con una *media* de 0.8 y 7 respectivamente.

6.3 YAZ

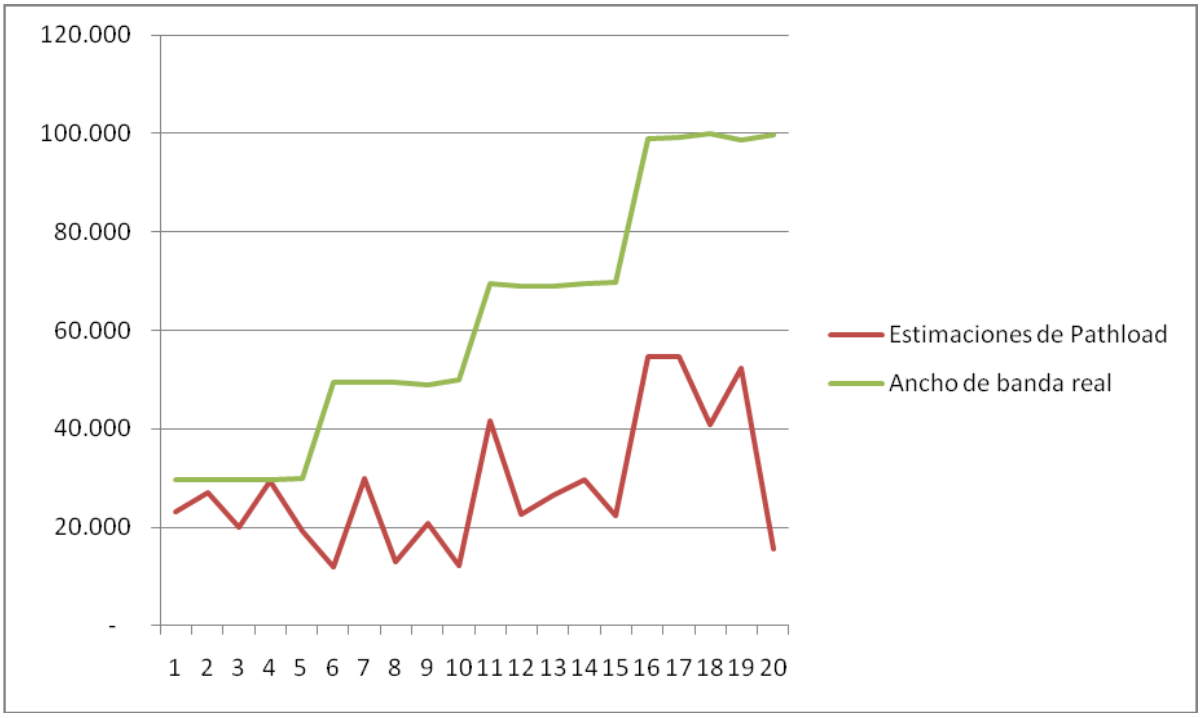
Yaz es una herramienta muy inestable y en muchos de sus resultados cuenta con un intervalo de confianza bien grande debido a su inestabilidad a la hora de hacer las estimaciones puntuales. La herramienta pide ser calibrada en muchas ocasiones cuando comienza a haber congestión en la red, pero aunque se calibre a la medida que sugiere, se repiten los experimentos y aún sigue pidiendo ser calibrada a otro o al mismo valor antes pedido.

El *overhead* y *tiempo de estimación* para esta herramienta es de los más pequeños, con una media alrededor de 0.4 y 1 respectivamente para todos los experimentos realizados.

6.4 PATHLOAD

Pathload es una herramienta que va mejorando a medida la congestión aumenta en el canal como se puede mostrar en la Figura 48. Su intervalo de confianza permanece muy similar en todos los experimentos.

Figura 48. Estimaciones de Pathload con tráfico Periódico



El *overhead* y *tiempo de estimación* para esta herramienta es de los más altos y con un intervalo de confianza bien grande, su media se sitúa alrededor de 2 y 8 respectivamente para la mayoría de los experimentos realizados.

6.5 SPRUCE

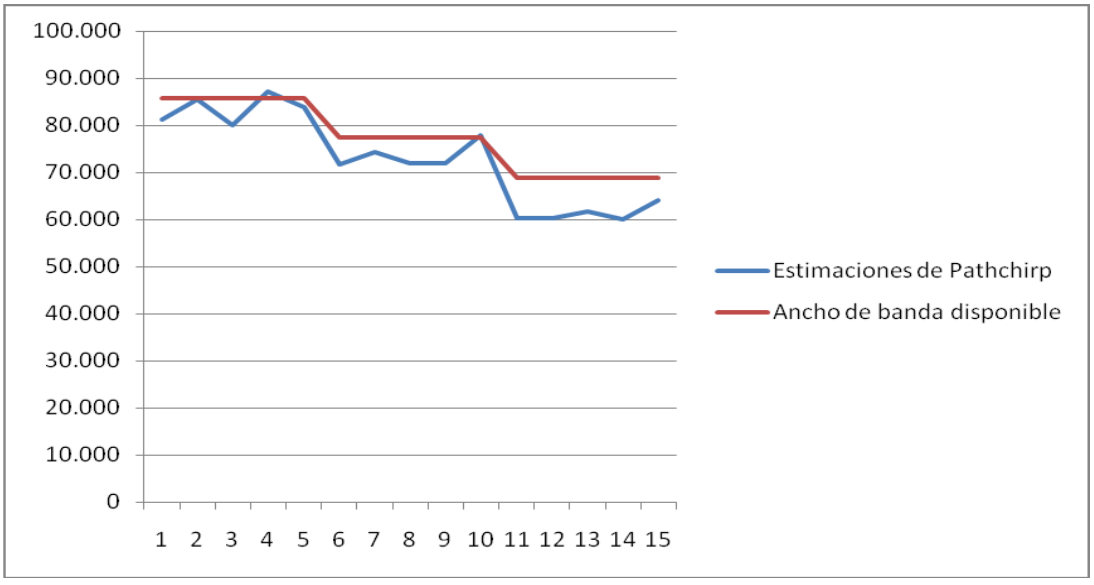
Spruce es una herramienta que a medida la congestión aumenta en el canal sus resultados comienzan a ser más erráticos como se ha mostrado en las Figuras de *Estimaciones* con los diferentes tipos de tráfico. Su intervalo de confianza permanece muy similar en todos los experimentos.

El *overhead* para esta herramienta es de los más pequeños y su intervalo de confianza también, su media se sitúa alrededor de 0.4 para la mayoría de los experimentos realizados. Pero a la vez es la herramienta que más demora para hacer las estimaciones, con una media alrededor de 11s.

6.6 PATHCHIRP

Pathchirp es la herramienta con mejores resultados para el tipo de experimento con congestión *Burst* como se muestra en la Figura 49. Estuvo entre las mejores en los experimentos realizados sin tráfico, debido a que su media se sitúa muy cercana al cero y su intervalo de confianza es bien pequeño. Su intervalo de confianza es de los más pequeños en general para todos los experimentos.

Figura 49. Estimaciones de Pathchirp con tráfico Burst

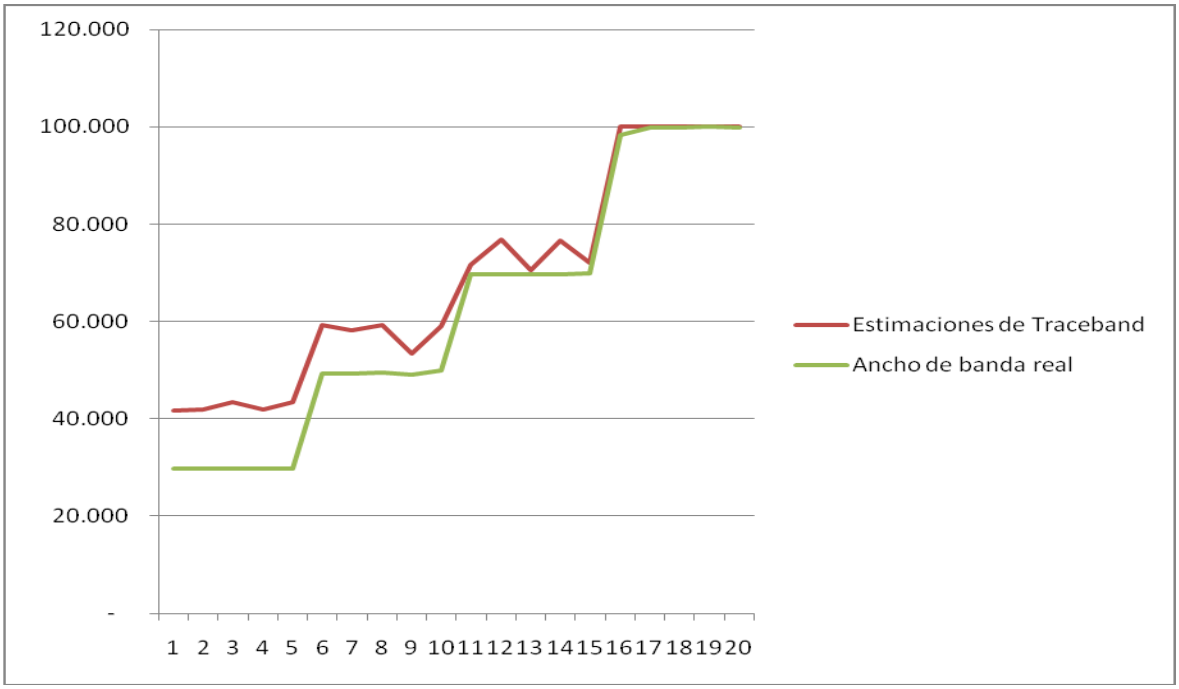


El *overhead* y *tiempo de estimación* para esta herramienta es de los más pequeños, además cuenta con un intervalo de confianza bien pequeño, sus medias se sitúan alrededor de 0.1 y 1s respectivamente para la mayoría de los experimentos realizados.

6.7 TRACEBAND

Traceband es la herramienta con más estabilidad en todas las pruebas, debido a su bajo nivel de error en la mayoría de ellas. Fue tomada como la mejor herramienta en 4 tipos de escenarios y en el resto muestra un comportamiento muy cercano al valor real. Su estabilidad hace de ella la herramienta más confiable. Por otra parte se observo que mientras la congestión aumenta en el canal sus resultados se van alejando más del valor real, como se muestra en la Figura 50.

Figura 50. Estimaciones de Traceband con tráfico Periódico.



Junto con *Pathload* es la herramienta con más alto índice de *overhead,* pero su *tiempo de estimación* es el más pequeño en todas las pruebas.

7. RECOMENDACIONES

Aplicar un *MovingAverage* a los resultados discretos que se tienen en cuenta a la hora de mostrar una estimación para así suavizar el *error de estimación*. Esto permitirá que las herramientas puedan mostrar resultados más cercanos a la realidad, debido a la inestabilidad que pueden tener entre sus estimaciones puntuales

Tratar de controlar los retardos de tiempo entre paquetes en cola en el procesador de tareas, ya que las herramientas que utilizan el *probe rate model* (*Pathload, Pathchirp y Yaz*) se basan en ello para hacer las estimaciones y requieren de q ue sus procesos sean atendidos una vez que lleguen a la cola de procesamiento.

BIBLIOGRAFÍA

A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, K. Claffy, Comparison of public end-to-end bandwidth estimation tools on high-speed links, in: Proceedings of the 6th Passive and Active Measurements Workshop, 2005.

B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estimating bandwidth bottlenecks, in: Proceedings of the IEEE Global Telecommunications Conference, vol. 1, San Francisco, CA, USA, 2000, pp. 415–420.

C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure?, in: Proceedings of the IEEE Infocom, 2001.

Guerrero Cesar, Labrador Miguel, Traceband: A fast, low overhead and accurate tool for available bandwidth estimation and monitoring. Internet Network Management Workshop (INM 2008). October 2008. Orlando, FL.

J. Sommers, P. Barford, W. Willinger, A proposed framework for calibration of available bandwidth estimation tools, Microprocessors and Microsystems 31 (2007) 222–235

J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of the 3rd ACM SIGCOMM conference on Internet Measurement, 2003, pp. 39–44.

L. Angrisani, S. DAntonio, M. Vadursi, G. Ventre, Performance comparison of different techniques for available bandwidth measurement in packet switched networks, in: VECIMS 2003 – International Symposium on Virtual Environments, Human–Computer Interfaces, and Measurement Systems, 2003.

M. Jain, C. Dovrolis, Pathload: A measurement tool for end-to-end available bandwidth, in: Proceedings of the 3rd Passive and Active Measurements Workshop, vol. 11, 2002, pp. 14–25.

Michaut Fabien, Lepage Francis, CRAN CNRS UMR7039, Application-Oriented Network Metrology: Metrics And Actives Measurement Tools.

M. Jain, C. Dovrolis, End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput, IEEE/ ACM Transactions on Networking 11 (4) (2003) 537–549.

Ns-2 The Network Simulator NS 2, Home page. Página principal. http://www.isi.edu/nsnam/ns/. Junio 2010.

Navratil, J., Cottrell, R.L.: ABwE: A Practical Approach to Available Bandwidth. In: Proceedings of the 4th International workshop on Passive and Active network Measurement PAM 2003 (2003)

N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal on Selected Areas in Communications 21 (6) (2003) 879–894.

S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, R. Fonseca, Measuring bandwidth between PlanetLab nodes, in: 6th Passive and Active Measurements Workshop, 2005.

V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, L. Cottrell, pathchirp: Efficient available bandwidth estimation for network paths, in: Proceedings of the 4th Passive and Active Measurements Workshop, vol. 2, 2003.