ANÁLISIS DEL DESEMPEÑO DE LA TÉCNICA DE ACCESO EN LAS REDES DE ÁREA CORPORAL. ANEXO.

Martha Isabel Mosquera Uribe
Camilo José Solarte Paz

Director
Ing. Oscar J. Calderón C.

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Línea de Investigación Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Enero de 2013

ANÁLISIS DEL DESEMPEÑO DE LA TÉCNICA DE ACCESO EN LAS REDES DE ÁREA CORPORAL. ANEXO.

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Martha Isabel Mosquera Uribe
Camilo José Solarte Paz

Director
Ing. Oscar J. Calderón C.

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Línea de Investigación Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Enero de 2013

ANEXO

INSTALACIÓN DEL SOFTWARE OMNeT++ Y DE LA HERRAMIENTA DE SIMULACIÓN CASTALIA

A. INSTALACIÓN DE OMNeT++ Y CASTALIA EN UBUNTU LINUX 10.04 [1]

Inicialmente se procede a descargar los archivos binarios del software OMNeT++ correspondiente a la versión 4.2.2. Para esto se accede a la página http://omnetpp.org/component/docman/cat_view/1-omnet-releases y se selecciona el paquete "*OMNeT++ 4.2.2 (source + IDE, tgz)".*

Antes de proceder con la instalación del software, se requieren instalar ciertos archivos previamente, entre los cuales se encuentran; el compilador C++ (gcc), la máquina virtual de JAVA y otros programas adicionales. Para ello se ingresa a la terminal del sistema como usuario *root* y se realiza la instalación por medio de los siguientes comandos:

```
$ sudo apt-get update

$ sudo apt-get install build-essential gcc g++ bison flex perl \
  tcl-dev tk-dev blt libxml2-dev zlib1g-dev openjdk-6-jre \
  doxygen graphviz openmpi-bin libopenmpi-dev libpcap-dev
```

Para continuar, se copia el paquete de OMNeT++ en el directorio donde se desee instalar y posteriormente se descomprime empleando la siguiente línea:

```
$ tar xvfz omnetpp-4.2.2-src.tgz
```

A continuación, se debe añadir el directorio *bin* a la ruta donde se encuentra instalado el software, con el fin de fijar el entorno de variables de manera permanente. Para esto se edita el archivo *.bashrc* alojado en el directorio *home*, haciendo uso del comando *gedit*, como se muestra a continuación:

```
$ gedit ~/.bashrc
```

Una vez abierto el archivo, se escribe la siguiente línea al final del mismo, se guardan los cambios y se reabre la terminal para que los cambios tengan efecto.

```
$ export PATH=$PATH: ~/omnetpp-4.2.2/bin
$ export LD_LIBRARY_PATH=~/omnetpp-4.2.2/lib
```

Finalmente se realiza la configuración y compilación del sistema, para lo cual se ingresa por la terminal al directorio donde se encuentra OMNeT++ y se digita lo siguiente:

```
$ NO_TCL=1 ./configure
$ make
```

De esta forma, ya se encuentra instalado el software OMNeT++ y se procede a descargar el código fuente de la versión 3.2 del Simulador Castalia desde la página *http://castalia.npc.nicta.com.au/new_view.php?nid=14*. Una vez finalizada la descarga, se descomprime el paquete desde la terminal por medio del siguiente comando:

$ tar –xvzf Castalia-3.2.tar.gz

Para continuar se procede a realizar la compilación utilizando las siguientes líneas:

$ cd Castalia-3.2/
$ ./makemake
$ make

Seguidamente, se verifica que se haya creado el directorio *CastaliaBin* dentro de la carpeta de Castalia, lo cual indica que la compilación ha sido exitosa. Antes de correr cualquier simulación se sugiere añadir la carpeta *Castalia/bin/* al entorno de variables, para lo cual se utiliza el siguiente comando:

$ export PATH=$PATH:~/Castalia/bin

B. MANUAL DE CASTALIA VERSIÓN 3.2 [2]

Castalia cuenta con *scripts*[1] [3] que permiten por medio de diferentes opciones, llevar a cabo tareas específicas. Estos se presentan a continuación:

- ***Castalia:*** se utiliza para definir una variedad de configuraciones y ejecutar diferentes pruebas de simulación. Las opciones disponibles se describen enseguida:

  -c CONFIG, --config=CONFIG
  Permite ejecutar una simulación con base en una lista de configuraciones permitidas. Para ello, se escriben los nombres de dichas configuraciones separadas por comas o se listan dentro de corchetes.

  -i FILE, --input=FILE
  Se utiliza para seleccionar el archivo de origen con que se ejecutará la simulación. Por defecto es omnetpp.ini.

  -o FILE, --output=FILE
  Permite definir o seleccionar el archivo de salida que contendrá los resultados. Por defecto este lleva como nombre, la fecha y la hora de creación.

  -r N, --repeat=N
  Indica el número de repeticiones que se ejecuta la simulación.

  Ejemplo 1:

  root@camilo-laptop:/home/camilo/Castalia-3.2/Simulations/BANtest# ../../bin/Castalia -c ZigBeeMAC,allNodesVaryPower,General -o prueba1.txt -r 5

  En la línea anterior, se ejecutan al mismo tiempo algunas de las configuraciones permitidas, estas se describen a continuación:

  ZigBeeMAC: simula una red de sensores que emplea el protocolo 802.15.4.

[1] *Script*: es un programa escrito para un software específico, que realiza automáticamente la ejecución de tareas, sin necesidad de ser ejecutadas una a una por el usuario.

AllNodesVaryPower: establece diferentes valores de potencia de transmisión a los nodos.

General: define las pérdidas de trayecto en los enlaces y variaciones temporales en el canal.

Por defecto se utiliza *omnet.ini* como archivo de entrada, se establece *prueba1.txt* como archivo de salida y se corre la simulación con cinco repeticiones.

Ejemplo 2:

Castalia –c ZigBeeMAC,allNodesVaryPower,[General,noTemporal] –o prueba2.txt –r 10

En este caso las configuraciones se intercalan con las contenidas en los corchetes, es decir que se ejecutan dos pruebas, la primera se corre utilizando *General* y la segunda *noTemporal*. En esta última no se simulan efectos de variaciones temporales en el canal.

- ***CastaliaResults:*** permite analizar, resumir y presentar el archivo de resultados generado por *Castalia*.

A manera de ejemplo se muestra la figura 1, en la cual se utiliza el *script* para analizar el archivo *prueba1.txt* obtenido anteriormente.

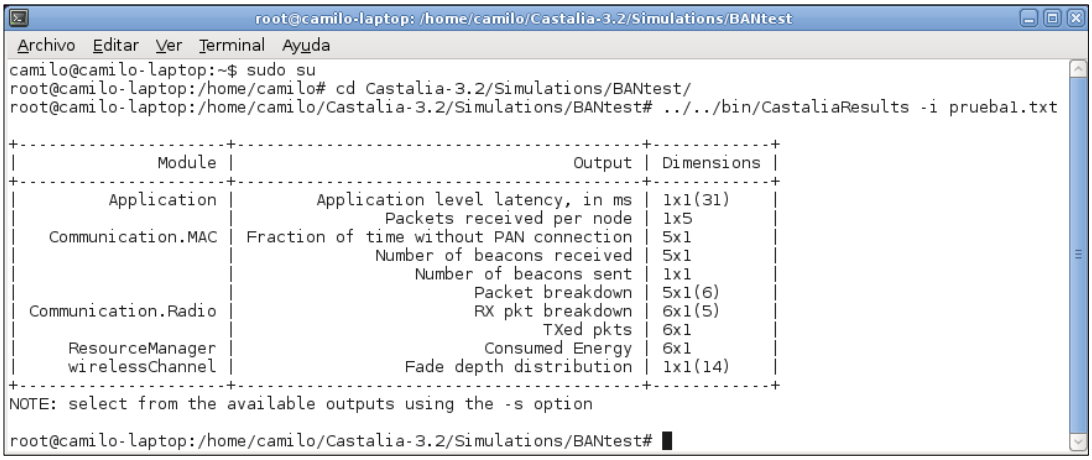

Figura 1. Análisis de un archivo por medio del *script CastaliaResults.*

La figura anterior muestra las salidas de los diferentes módulos del archivo de resultados, las cuales pueden ser presentadas en tablas numéricas, utilizando la siguiente opción:

-s ER, --show=ER(*Expresión Regular*)

La *"expresión regular*[2]*"* [4] se refiere a un nombre que se compara con los caracteres de las salidas del archivo, una vez coincidan se presentan los resultados correspondientes, como lo muestra la figura 2.

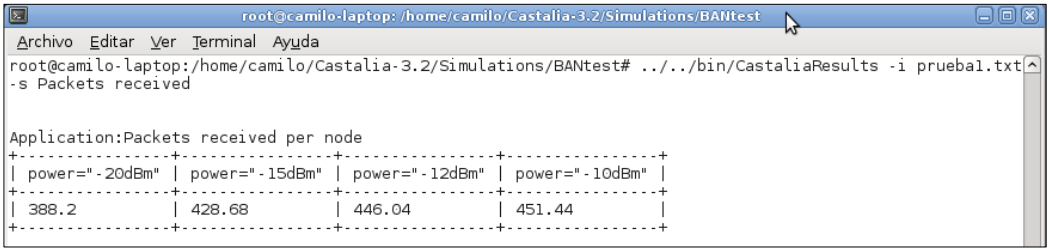

Figura 2. Presentación de los resultados para la salida *Packets received*.

Otras de las opciones más útiles del *script*, se describen a continuación:

-f ER, --filter-rows=ER –filter-columns=ER
Filtra únicamente los resultados de las filas o de las columnas de la tabla respectivamente, que correspondan con la ER.

-n –node
Despliega los resultados para cada nodo individualmente.

-p, --percentaje
Muestra los resultados como porcentajes de la cantidad total.

--precision=N
Define la precisión de los números de punto flotante en decimales.

--sum
Obtiene como resultado la suma de todos los valores, en vez del promedio de los mismos.

- ***CastaliaPlot:*** este *script* permite graficar los datos obtenidos por *CastaliaResults*. Se destacan las siguientes opciones:

  -g, --greyscale
  Grafica la figura únicamente en escala de grises.

  -l Legend, --legend=LEGEND
  Fija la ubicación de la leyenda en la figura.

  -o FILE, --output=FILE
  Selecciona o define el archivo de salida.

  -s STYLE, --style=STYLE

[2] *Expresión regular:* se utiliza para emparejar cadenas de texto, tales como palabras, caracteres particulares o patrones de caracteres.

Define el estilo con que se grafica la figura, esta puede ser un histograma, de barras apiladas o de puntos y líneas.

--colors=COLORS
Establece el color de la figura. Para obtener una lista completa de los colores disponibles se utiliza el comando –-colors=?.

--hist-box=BOXWIDTH
Fija el ancho de las columnas en un histograma.

--hist-gap=HISTGAP
Fija el espacio entre las columnas en un histograma.

--invert
Invierte la tabla recibida como entrada por el *script*, al momento de realizar la figura, es decir que toma las filas como columnas y viceversa.

--linewidth=LINEWIDTH
En el estilo de puntos y líneas, establece el ancho de las mismas.

--order-columns=ORDER
Se utiliza para ordenar u omitir las columnas de una tabla en la figura. Este comando debe ser aplicado antes de invertir.

--xrotate=ROTATE
Se utiliza para rotar los valores del eje X.

--xtitle=XTITLE --ytitle=YTITLE
Define el título del eje X y del eje Y respectivamente.

--yrange=MIN:MAX
Establece el rango de valores que toma el eje Y en la figura.

Ejemplo 3:

En este ejemplo se utiliza *CastaliaResults y CastaliaPlot* para obtener la figura 3, esta muestra los paquetes recibidos por nodo, utilizando el archivo *prueba1.txt*. Los comandos necesarios y la figura en cuestión se presentan enseguida:

root@camilo-laptop:/home/camilo/Castalia-3.2/Simulations/BANtest#
../../bin/CastaliaResults -i prueba1.txt -s Packets received -n | ../../bin/CastaliaPlot -o figura1.png -s histogram --hist-box=0.6 --hist-gap=0 --xtitle=TRAMAS/RECIBIDAS/POR/NODO/PRUEBA_1 --ytitle=NUMERO/DE/TRAMAS --xtitle=NODO -l "top left"

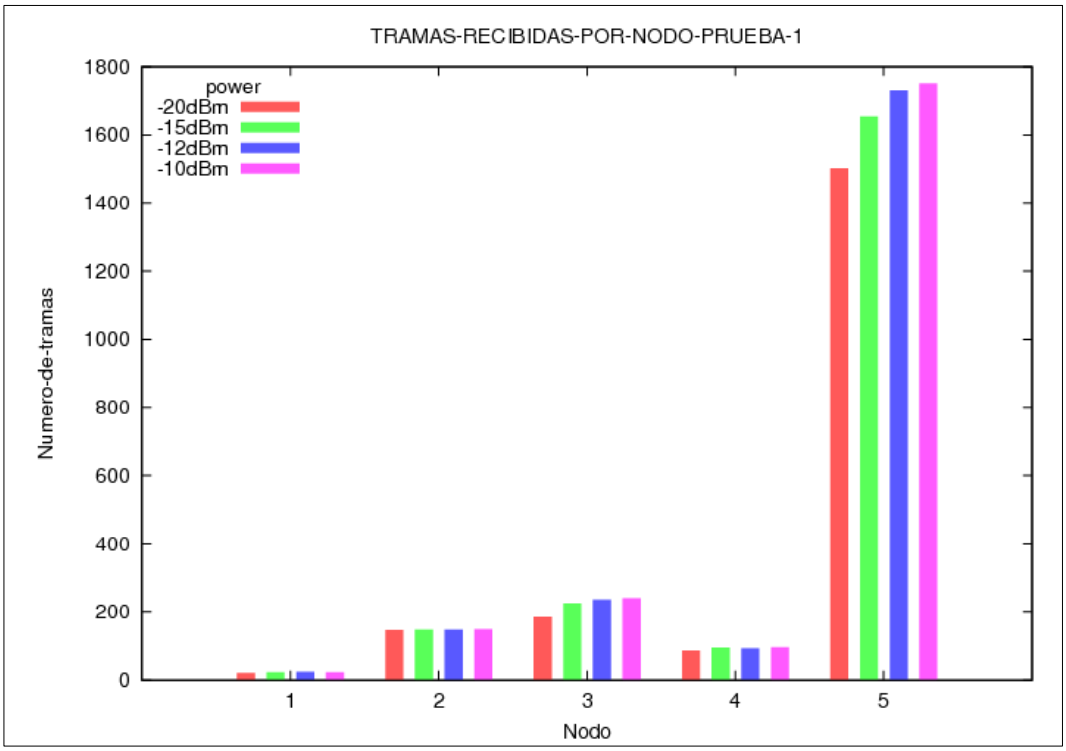

Figura 3. Despliegue de resultados empleando *CastaliaPlot*.

El simulador Castalia utiliza en conjunto los tres *scripts* descritos anteriormente, con los cuales es posible realizar la configuración de las pruebas, la interpretación de los resultados y la visualización de los mismos.

REFERENCIAS

[1] NICTA. (2011, Dec 18 2012). *How To Install Castalia*. Available: http://castalia.npc.nicta.com.au/pdfs/Castalia%20-%20Installation.pdf

[2] A. Boulis. (2011, Oct 28 2012). *A simulator for Wireless Sensor Networks and Body Area Networks, version 3.2* [User´s manual]. Available: http://castalia.npc.nicta.com.au/pdfs/Castalia%20-%20User%20Manual.pdf

[3] R. P. Loui, "In Praise of Scripting: Real Programming Pragmatism," *Computer,* vol. 41, pp. 22-26, 2008.

[4] K. Thompson, "Programming Techniques: Regular expression search algorithm," *Communications of the ACM,* vol. 11, pp. 419-422, 1968.