Simulaciones Gráficas de redes Bluetooth utilizando el Network Animator del NS2 con la librería UCBT

Ing. Francisco Novillo, José Landa A., José Jimbo S. y Luis Salazar M.
Facultad de Ingeniería en Electricidad y Computación, ESPOL, Guayaquil-Ecuador
Grupo de Investigación en Comunicaciones Móviles (GICOM)
{fnovillo, jlanda, jjimbo, lsalazar}@fiec.espol.edu.ec

Abstract.-**La visualización gráfica de resultados dentro de las operaciones que siguen los dispositivos Bluetooth dentro de su estructura básica y sus procedimientos, tales como el descubrimiento, conexión y transmisión de archivos, nos ayudan a tener en cuenta de manera mas didáctica el funcionamiento de esta tecnología de corto alcance. Utilizando herramientas como el simulador Network Simulator, su extensión gráfica Network Animador y la librería para el uso de Bluetooth de la Universidad de Cincinnati UCBT, hemos logrado adaptarla para la interpretación gráfica de las simulaciones Bluetooth.**

***Index Terms*— Ad-Hoc, Bluetooth, NS2, NAM, UCBT.**

I. INTRODUCION

Bluetooth es un estándar de comunicación inalámbrica de corto alcance, bajo costo y bajo consumo de energía, aún en desarrollo, para aplicaciones de corto alcance. Dentro de lo que es Bluetooth se tiene sus formas básicas de conexión de dispositivos como lo son las piconets y las scaternets. En el estándar se han definido los parámetros de cómo debe trabajar y sus procedimientos básicos de interconexión de dispositivos así como de los perfiles para la interoperabilidad entre varios fabricantes. Dado que es una tecnología que aún sigue en desarrollo, una herramienta poderosa para poder probar diseños, algoritmos de ruteo en redes Ad Hoc, lo constituye el simulador de redes NS2. Dentro de las herramientas para la visualización tenemos las desarrolladas como el NAM (Network Animador) y el Xgraph (para ver resultados en graficas). Nuestro trabajo se deriva del desarrollo de la librería para la simulación de Bluetooth de la universidad de Cincinnati UCBT, en síntesis esta funciona con el NS2, pero hemos planteado una adaptación para que trabaje con el NAM, dándole un ambiente gráfico más completo, para observar los procesos que se dan dentro de las operaciones de los dispositivos Bluetooth.

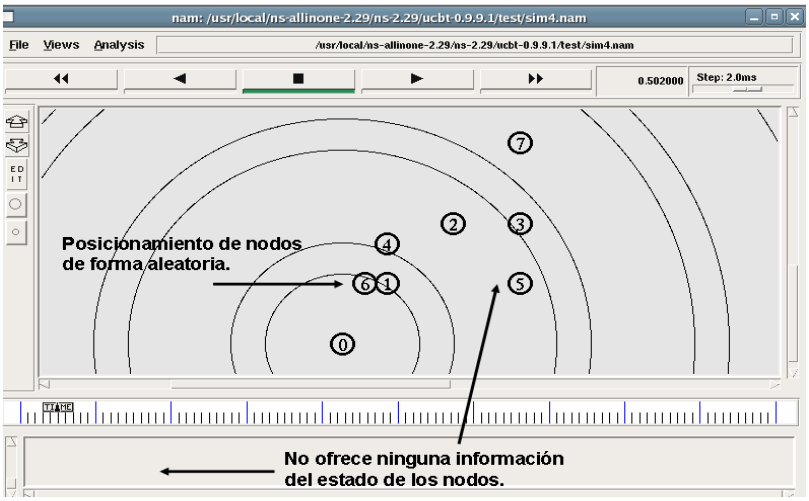
El propósito de esta adaptación gráfica de UCBT es poder tener una herramienta fiable de simulación de varios casos dentro de la tecnología Bluetooth así como el poder hacer análisis y contribuciones futuras.

II. SOLUCIÓN PLANTEADA

Para solucionar el principal problema que presenta la librería UCBT, el cual es, que actualmente carece de una interfaz gráfica que pueda ser manipulada por el usuario, decidimos incorporar a la librería, en las funciones dentro de las cuales esta nos permita realizar las principales acciones tales como el posicionamiento y el movimiento de los nodos, así como el código necesario para que durante la simulación en el NAM (Network Animator) estas puedan ser visualizadas.

Además para revisar el estado de los nodos en un instante de tiempo tenemos que recurrir al archivo de salida del simulador, en otras palabras, el simulador no ofrece información del tiempo en que ocurre el cambio de estado de los nodos y cuales son estos.

Lo que planteamos como solución es realizar modificaciones a ciertas funciones que nos permitan realizar acciones tales como movimiento y posicionamiento ya que la versión de la librería UCBT que utilizamos no ofrece tales opciones, como podemos observar en la Fig. 1.1 el posicionamiento de los nodos es de forma fija para todas las simulaciones que se realicen, de igual manera no se presenta ninguna información acerca del estado de los nodos durante la simulación haciendo

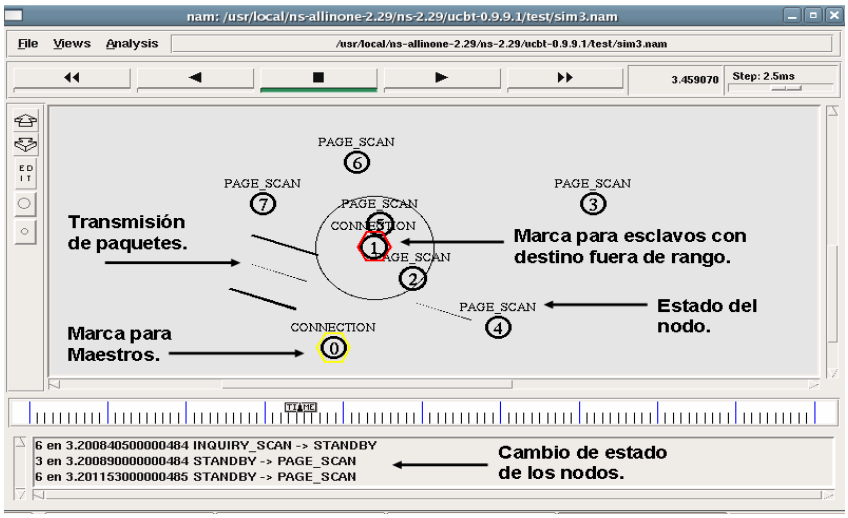


la interfaz grafica de poca utilidad para el aprendizaje.

Para solucionar estos problemas decidimos incorporar a la parte gráfica varia información durante la simulación, entre

una de las mas importantes es la información acerca del estado del nodo puesta sobre cada uno de ellos.

Como aportes adicionales mostramos como comentario información del nodo, como lo es la identificación, el instante de tiempo en el cual este cambia del estado y el estado anterior y actual del mismo. Además se le colocaran marcas (Amarillas) a los nodos que alcanzan el estado de Master (Maestro) y marcas (Rojas) a los nodos móviles cuyo destino se encuentre fuera del área de cobertura para obtener una mejor apreciación de los resultados en cada instante de tiempo durante la duración de la simulación, los resultados después de las modificaciones realizadas a la librería se pueden apreciar en la Fig. 1.2.



III. DESCRIPCIÓN DE LAS MODIFICACIONES A LA LIBRERÍA UCBT.

Para realizar las modificaciones a la librería y obtener los resultados deseados usaremos dos herramientas, estas son:

1. La función ***tcl.eval (char* x)*** **[1]** de la clase Tcl instance que esta declarada en ***~tclcl/Tcl.cc***, para obtener acceso a esta instancia debemos declarar la siguiente línea en cada función de la librería donde se use ***tcl.eval (char* x)***.

   Tcl& tcl = Tcl :: instance ();

2. El formato de las trazas del archivo del NAM (Network Animator), para cada evento **[2] [3]**.

Utilizaremos estas herramientas en ciertas funciones que se encuentran dentro de algunos archivos que tienen implementadas funciones especificas de ***baseband, wirelessnode y ns-btnode*** estas funciones fueron seleccionadas por ofrecernos información acerca del estado, los posicionamientos y los tiempos en que ocurren los cambios de estados en cada nodo así como también la identificación del nodo que en un tiempo determinado esta siendo tratado.

La mayor cantidad de código se añadió en el archivo en el que se especifican las funcionalidades de ***baseband***, pues es aquí donde se realizan la mayor cantidad de eventos dentro de la librería UCBT. Además del código añadido se declararon dos variable globales estas son ***Coor [50] [8]*** de tipo double y ***sec*** de tipo *entero*, el funcionamiento de cada una de estas variables de explicarán mas adelante.

La variable ***Coor [50] [8]*** es una matriz que acepta un número máximo de 50 nodos por defecto y almacena importante información como lo son las coordenadas de los nodos y además banderas utilizadas para identificar ciertos estados de los nodos como lo son si es Maestro, el nodo esta marcado, el nodo esta conectado, etc. Esta información servirá para poder realizar acciones dependiendo del caso en las funciones que se han modifica dentro de las funcionalidades del ***baseband,*** en la Fig. 1.3 se muestra la matriz definida por la variable ***Coor [50] [8]***. Las filas de la matriz representa el identificador del nodo aprovechando el hecho que en la mayoría de simulaciones se crean nodos de manera secuencial, de esta manera asociamos las filas de la matriz con el identificador de cada nodo es por eso que en la mayoría de veces usaremos la variable ***Coor [ ] [ ]*** de esta manera ***Coor [bd_addr_] [ ], Coor [sender] [ ]*** y ***Coor [receiver] [ ]***, siendo bd_addr_, sender, receiver las variables definidas por la librería UCBT en la seccion del ***baseband*** como identificadores de los dispositivos.

Los datos almacenados en las localidades de la columna ***Pos X*** (***Coor [ ] [0]***) representan las posiciones en x de los nodos, si hacemos la analogía con las coordenadas (x, y), de igual forma los datos almacenados en las localidades de la columna ***Pos Y*** (***Coor [ ] [1]***) representan las posiciones en y del nodo.

Los datos almacenados en la columna ***f Coor*** (***Coor [ ] [2]***) son banderas que nos indican si esta en 0 que el nodo no se ha movido y si esta en 1 que este ya ha realizado por lo menos un movimiento. Usamos esta bandera debido a que la librería UCBT maneja una estructura llamada ***node_*** en la cual se encuentras definidas las variables ***X_*** (***node_->X_***), ***Y_*** (***node_->Y_***) y ***Z_*** (***node_->Z_***) estas variables guardan información de la posición de los nodos pero cuando se realiza el primer movimiento estos valores son distintos a la posición de destino del nodo, así que a partir del primer movimiento estos valores no son útiles como posición de partida para un nuevo movimiento del mismo, si usamos estas variables para otro movimiento se creara efectos de brinco al involucrarse la función que realiza el efecto de la movilidad.

Por ello el primer valor que almacenara ***Coor [ ] [0]*** y ***Coor [ ] [1]*** serán los valores que contengan las variables ***node_->X_*** y ***node_->Y_***, la variable ***node_->Z_*** es descartada porque no

	Pos X	Pos Y	f Coor	f Mark MST	f Mark OUTR	f es MST	f MSTid	f CONNET
0	0,00	0,00	1	0	0	1	0	0
1	5,00	2,36	0	0	1	0	0	1
2	10,50	25,36	0	1	0	1	0	1
3	1,00	0,00	0	1	1	0	3	1
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
47	25,69	14,82	1	0	0	0	6	1
48	12,36	1,65	0	0	1	1	9	1
49	13,54	2,54	1	1	1	0	5	1

trabajamos en tres dimensiones, cuando se realiza el primer movimiento los datos almacenados en ***Coor [ ] [0]*** y ***Coor [ ] [1]*** serán la posición de destino del nodo, es por ello que antes de asignar el valor a estas variables verificamos el estado de ***f Coor*** (***Coor [ ] [2]***).

Los datos almacenados en la columna ***f Mark MST*** (***Coor [ ] [3]***) son banderas que nos indican si el nodo es Master (Maestro) y es usada para ponerle una marca que lo distinga del resto de nodos, esta marca es un hexágono de color amarillo colocado sobre el nodo, esto ocurre cuando el valor de esta bandera para el correspondiente nodo es 1.

Cuando el nodo sale del estado de Maestro entonces esta bandera toma el valor de 0 y la respectiva marca que lo identificaba como tal debe ser removida.

Los datos almacenados en la columna ***f Mark OUTR*** (***Coor [ ] [4]***) son banderas usadas cuando realizamos algún movimiento de nodos, se activa cuando el nodo móvil realiza un movimiento cuyo destino se encuentra fuera del rango de la piconet a la que pertenece, caso contrario permanece en 0, esto nos es útil para colocarle una marca a ese nodo, esta marca es un hexágono de color rojo colocado sobre el nodo, cuando se ha activado esta bandera y para poder diferenciarlo del resto de igual forma si el mismo nodo realiza otro movimiento cuyo destino este dentro de rango de la piconet en la que se encuentra, la marca que este poseía será quitada.

Los datos almacenados en la columna ***f es MST*** (***Coor [ ] [5]***) son banderas usadas para indicarnos si el nodo es maestro en el caso de estar esta activa, con el valor de 1, y que no lo es cuando esta tiene el valor de 0.

Los datos almacenados en la columna ***f MSTid*** (***Coor [ ] [6]***) en cambio nos indican la piconet a la cual pertenecen estos nodos y lo hacemos asociando al nodo con el identificador del Maestro al cual están conectados.

Los datos almacenados en la columna ***f CONNET*** (***Coor [ ] [7]***) son banderas que nos indican si el nodo es un esclavo en caso de estar activa con el valor de 1 y que no lo es cuando tiene el valor de 0, su funcionamiento en el equivalente al de ***f es MST*** pero para el caso de esclavos.

Para realizar las acciones que se visualizarán en NAM (Network Animator) hacemos uso de los datos en la matriz ***Coor [50] [7]*** mediante validaciones y el uso de la función ***tcl.eval (char* x)*** para hacer que se ejecute la correspondiente acción, con del uso del correspondiente formato de las trazas que se almacenan en el archivo de NAM.

Para el caso de colocar los estados de los nodos sobre cada uno de ellos debemos usar el siguiente formato:
tcl.evalf ("puts $namfile \"n -t %.15f -s %d -S DLABEL -l %s -L %s\"", clkoffset, bd_addr_, nuevoest, lastest)

Donde ***clkoffset*** representa el instante de tiempo en que aparecerá esta acción durante la simulación, ***bd_addr_*** es el identificador del nodo, ***nuevoest*** es el nuevo estado del nodo y ***lastest*** el estado anterior.

Para colocar comentarios del instante en que el nodo cambia de estado y cuales son estos, primeros crearemos una variable tipo *string* que contenga el comentario que aparecerá en la ventana de NAM de la siguiente manera:

sprintf (anotacion, "%d en %.15f %s -> %s", bd_addr_, Scheduler::instance ().clock (), lastest, nuevoest)

Donde ***bd_addr_*** es el identificador del nodo, ***Scheduler::instance().clock()*** es el instante de tiempo en que ocurre el cambio de estado, ***sec*** es una variable global ya descrita anteriormente que actúa para generar la secuencia en la que aparecerán los comentarios en la pantalla del NAM, ***lastes*** y ***nuevoest*** son el nuevo estado y el estado anterior. Luego para ejecutar esta acción en el simulador usaremos el siguiente formato:

tcl.evalf ("puts $namfile \"v -t %.15f -e sim_annotation %.15f %d %s\"", clkoffset, clkoffset, sec, anotacion)

Cuando tengamos que ponerle las marcas (Amarillas) a los nodos que alcanzan el estado de maestro usaremos el siguiente formato:

tcl.evalf("puts $namfile \"m -t %.15f -s %d -n m%d -c yellow -h hexagon\"", clkoffset, bd_addr_, bd_addr_)

Y para remover estas marcas cuando estos nodos salgan del estado de maestros usaremos este formato:

tcl.evalf("puts $namfile \"m -t %.15f -s %d -n m%d -c yellow -h hexagon -X\"", clkoffset, bd_addr_, bd_addr_)

La única diferencia entre los dos formatos es el parámetros ***–X* [2] [3]**, este formato le indica al interprete del NAM que si un nodo tiene una marca puesta se la remueva.

el segmento de código agregado para la parte anteriormente descrita, la cual se encuentra en la función ***change_state()***, es descrita en el siguiente algoritmo:

```
change_state()
Tcl& tcl = Tcl :: instance ();
anotacion [100], nuevoest [100], lastest [100]
.
.
.
.
.
if (!strcmp(state_str(), “NEW_CONNECTION_SLAVE”)) then
        f es SLV [n] = 1
else
        if (f es SLV &&(!strcmp(state_str(), “CONNECTION”) then
                f es SLV [n] = 1
        else
                f es SLV [n] = 0
if (!strcmp(state_str(), “NEW_CONNECTION_MASTER”)
        sprintf(nuevoest, “NEW_CONNECTION_MST”)
        if (f Coor [n] ! = 1) then
                f Coor [n] = 1
                Pos X [n] = node_->X_
                Pos Y [n] = node_->Y_
        f es MST = 1
        if (f Mark MST [n] ! = 1 ) then
                f Mark MST [n] = 1
                .
                tcl.evalf("puts $namfile \"m -t %.15f -s %d -n
                m%d -c yellow -h hexagon\"", clkoffset,
                bd_addr_, bd_addr_)
else
        sprintf(nuevoest, “%s”, state_str())
        if(!strcmp(state_str(), “CONNECTION”) && f Mark
MST [n])
                f Mark MST [n] = 1
        else
                if (f Mark MST [n])
                        f Mark MST [n] = 0
                        .
                        tcl.evalf("puts $namfile \"m -t %.15f
                        -s %d -n m%d -c yellow -h hexagon
                        -X\"", clkoffset, bd_addr_,
                        bd_addr_)
if (!strcmp(ps, “NEW_CONNECTION_MASTER”))

        sprintf(lastest, “NEW_CONNECTION_MST”)
else
        sprintf (lastest, “%s”, ps)
sprintf (anotacion, "%d en %.15f %s -> %s", bd_addr_,
Scheduler::instance ().clock (), lastest, nuevoest)
.
.
.
tcl.evalf ("puts $namfile \"n -t %.15f -s %d -S DLABEL -l %s -
L %s\"", clkoffset, bd_addr_, nuevoest, lastest)
tcl.evalf ("puts $namfile \"v -t %.15f -e sim_annotation %.15f
%d %s\"", clkoffset, clkoffset, sec, anotacion)
```

Para lograr el posicionamiento inicial de los nodos, este es ejecutado cuando se invoca a la función ***setPos(X, Y)***, usaremos el siguiente formato:

tcl.evalf("puts $namfile \"n -t * -s %d -x %.17f -y %.17f -Z 0 -z 1 -v circle -c black\"", getAddr(), x, y)

Donde ***getAddr (), x, y*** son el identificador del nodo, la posición en x y la posición en y respectivamente.

Para realizar el efecto de movilidad, que es invocado con la función ***setdest ( )***, durante la simulación se utilizara el siguiente formato:

tcl.evalf ("puts $namfile \"n -t %.9f -s %d -x %.17f -y %.17f -U %.6f -V %.6f -T %.6f\"", clkaux, bd_addr_, Coox, Cooy, Vx, Vy, tmp)

Donde ***Coox*** y ***Cooy*** son las posiciones iniciales desde donde el nodo empezará a moverse, ***Vx*** y ***Vy*** son las velocidades en x y la velocidad en y respectivamente y ***tmp*** es el tiempo que dura el movimiento del nodo.

Ahora si el caso es poner una marca a los nodos que salgan del radio de la piconet en la que se encuentran utilizaremos el siguiente formato:

tcl.evalf ("puts $namfile \"m -t %.15f -s %d -n md%d -c red -h hexagon\"", clkaux, bd_addr_, bd_addr_)

Donde la variable ***clkaux*** cumple la misma función que la variable ***clkoffset*** anteriormente descrita. Para remover estas marcas utilizaremos igual que el caso anterior el siguiente formato:

tcl.evalf ("puts $namfile \"m -t %.15f -s %d -n md%d -c red -h hexagon -X\"", clkaux, bd_addr_, bd_addr_)

De igual forma la única diferencia entre los dos formatos anteriores es el parámetro ***–X* [2] [3]** utilizado para remover las marcas de un nodo que poseía una.

Podemos describir el segmento de código agregado a la función de movilidad mediante el siguiente algoritmo:

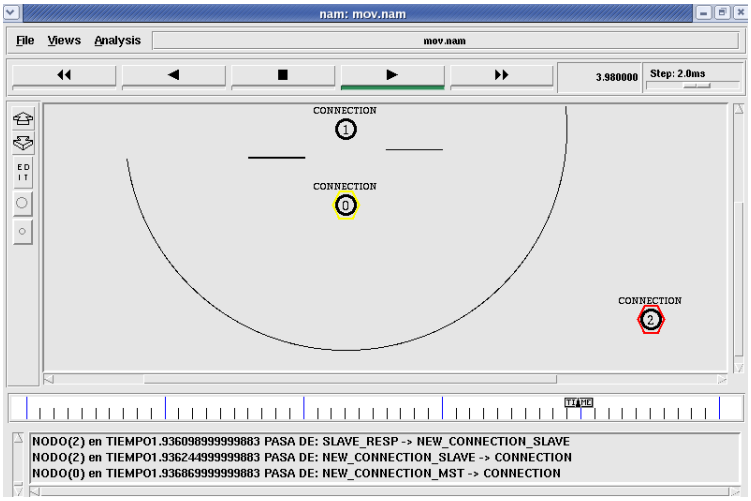
Setdest (destx, desty, destz, speed)

```
Tcl& tcl = Tcl :: instance ()
dst = 0, Coox = 0 ,Cooy = 0, Vx = 0, Vy = 0, tmp= 0, mst = 0,
dstmst = 0
.
.
if (f Coor [n] = 1) then
        Coox = Pos X [n]
        Cooy = Pos Y [n]
        Pos X [n] = destx
        Pos Y [n] = desty
else
        Coox = node_->X_
        Cooy = node_->Y_
        Pos X [n] = destx
        Pos Y [n] = desty
        f Coor [n] = 1
dst = sqrt((destx – Coox) (destx – Coox) + (desty – Cooy)
(desty – Cooy))
Vx = speed * (destx – Coox)/dst
Vy = speed * (desty – Cooy)/dst
tmp = dst/speed
.
tcl.evalf ("puts $namfile \"n -t %.9f -s %d -x %.17f -y %.17f
-U %.6f -V %.6f -T %.6f\"", clkaux, bd_addr_, Coox, Cooy,
Vx, Vy, tmp)
if (f es SLV [n] = 1) then
        if (f Mark OUTR [n] != 0) then
                mst = f MST id [n]
                dstmst = sqrt ((destx – Pos X [mst]) *
                (destx – Pos X [mst]) + (desty – Pos Y
                [mst]))
                if (dstmst > 10) then
                        f Mark OUTR = 1
                        tcl.evalf("puts $namfile \"m -t %.15f
                        -s %d -n md%d -c red -h
                        hexagon\"", clkaux, bd_addr_,
                        bd_addr_)
        else
                if (dstmst <= 10) then
                        f Mark OUTR = 0
                        tcl.evalf("puts $namfile \"m -t %.15f
                        -s %d -n md%d -c red -h hexagon -
                        X\"", clkaux, bd_addr_, bd_addr_)
```

IV. SIMULACIONES

En este trabajo presentamos tres simulaciones que realizamos con la adaptación grafica. La primera es un escenario con tres dispositivos bluetooth donde establecemos comunicación entre dos nodos esclavos y un maestro. Luego se efectúa el movimiento de un esclavo de tal manera que quede fuera del rango establecido por la especificación. Esto lo vemos en la Figura 1.4

Figura 1.4 Dispositivo Bluetooth fuera de rango de cobertura



La segunda simulación consiste en un caso típico de una Piconet. Tenemos cinco nodos bluetooth, cuatro esclavos y un maestro, estableciendo una transferencia de archivos entre los nodos 0 y 2 y los nodos 0 y 3. Los resultados pueden ser observados en la Figura 1.5

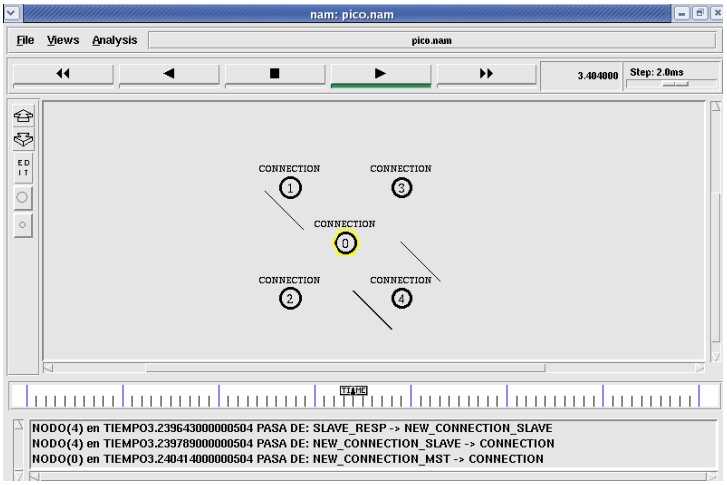


Figura 1.5 Transferencia de archivos en una piconet

En la tercera simulación tenemos una conexión TCP entre los nodos 0 y 2 (comunicación con acuse de recibo-ACK) y una conexión UDP entre los nodos 0 y 1 (sin acuse de recibo). Vea la Figura 1.6

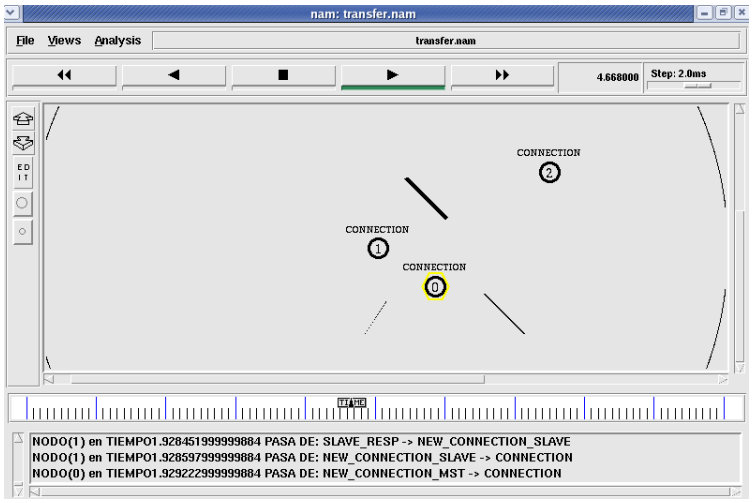


Figura 1.6 Tipos de transferencias en una red Bluetooth

V. CONCLUSIONES

En este trabajo, nosotros hemos introducido una solución práctica para facilitar la interpretación de los resultados obtenidos en las simulaciones realizadas con la librería UCBT.

Hemos mostrado que las modificaciones realizadas a ciertas funciones de la librería UCBT nos han permitido corroborar el correcto funcionamiento de la especificación BT como por ejemplo los estados por los que tiene que someterse un dispositivo para establecer una comunicación, la movilidad de dos dispositivos conectados para demostrar que el rango de cobertura no debe sobrepasar los diez metros y la transferencia de información entre un master y un esclavo.

Para mejorar el entendimiento de todo el proceso que se lleva a cabo para establecer una comunicación entre dos dispositivos BT, nuestras simulaciones deberán enfocarse en los procedimientos básicos tales como el Inquiry y Paging. Este será nuestro siguiente avance en nuestro trabajo.

VI. REFERENCIAS

[1] The ns Manual (formerly ns Notes and Documentation).
The VINT Project. A collaboration between researchers at UC Berkeley, LBL. USC/ISI, and Xerox PARC. Kevin Fall (kfall@ee.lbl.gov), Editor Kannan Varadhan (kannan@catarina.usc.edu), Editor January 8, 2003, Pag. 19.

[2] The ns Manual (formerly ns Notes and Documentation).
The VINT Project. A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Kevin Fall (kfall@ee.lbl.gov), Editor Kannan Varadhan (kannan@catarina.usc.edu), Editor January 8, 2003, Pag. 353.

[3] 38.1.11 Nam Trace File Format Lookup Table.
http://www-rp.lip6.fr/~ridoux/Docs/Manuel-NS2/node514.html, revisada el 1 de Agosto de 2006.

[4] UCBT Bluetooth Extension for NS2 at the University of Cincinnati
http://www.ececs.uc.edu/~cdmc/ucbt/ucbt.html, revisada el 15 de Enero de 2006.

[5] Bluetooth SIG
http://www.bluetooth.com/Learn/Technology/Specifications