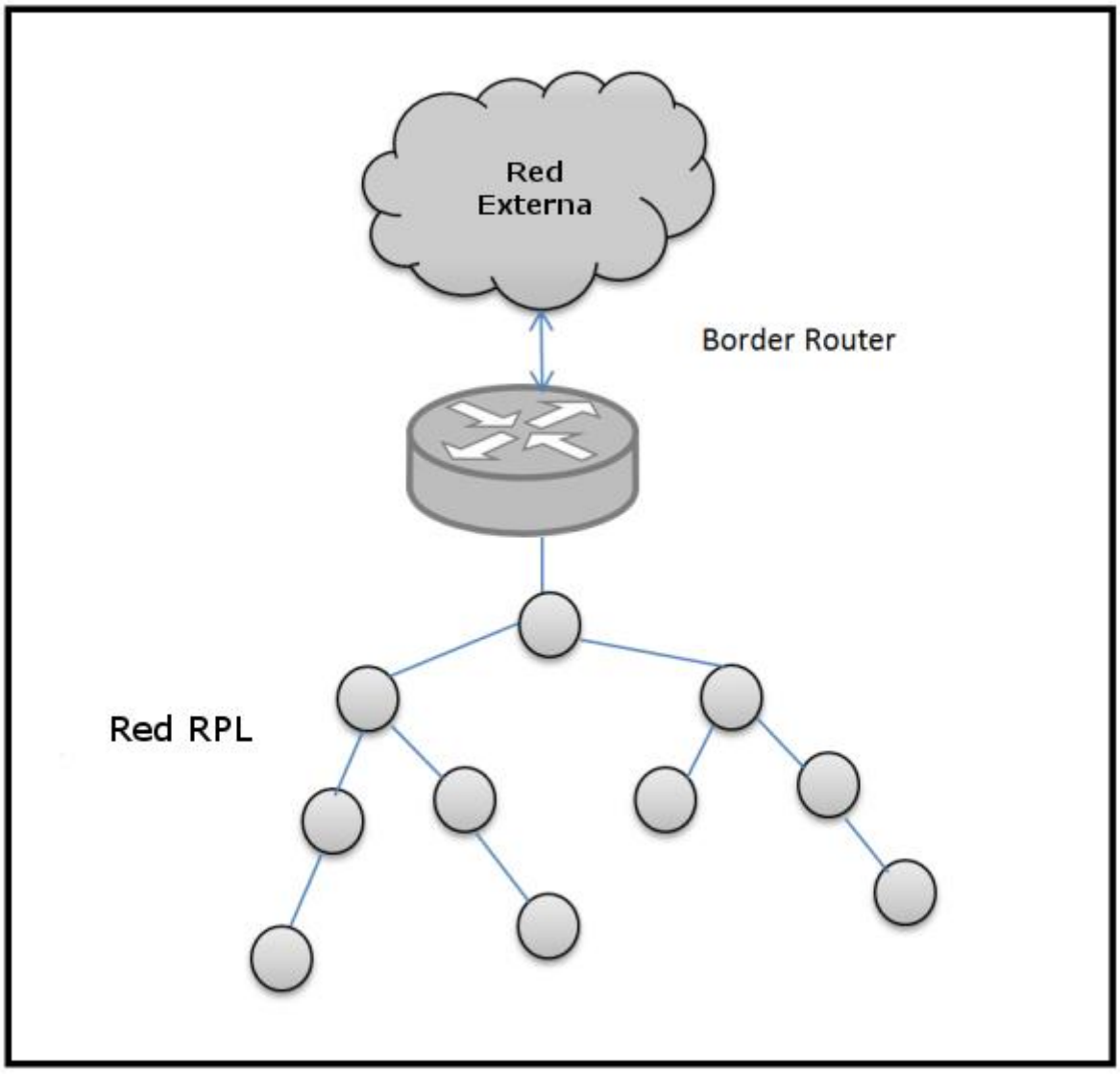
RPL Border Router

Introducción:

Los border routers son enrutadores que se encuentran en el borde de una red. Su función es conectar una red a otra, en este caso una red 802.15.4 a Internet.



Objetivo

El objetivo de este ejemplo es brindar una comprensión del funcionamiento de RPL y un Border Router en Contiki OS y aprender a simular una red con un border router en Cooja.

Bloques funcionales.

Los nodos *udp-client* formarán un DAG con el border router como raíz. El border router recibirá el prefijo a través de una conexión SLIP (Serial Line Interface Protocol) y se lo comunicará al resto de los nodos en la red RPL. Esto permite que los mismos se puedan acceder desde fuera de la red con ese prefijo de red.

Por default el border router aloja una página Web simple.

Simulación en Cooja

Es conveniente abrir tres terminales, una para iniciar Cooja, otra para establecer la comunicación con el border router con tunslip, y otra para comandos como ping. Inicie el simulador Cooja con la siguiente secuencia de commandos:

```
cd contiki-ng/tools/cooja
ant run
```

Cuando se inicie la GUI, ejecute los siguientes pasos para crear una simulación.

1. Desde 'File' seleccione 'New Simulation'. Seleccione 'UDGM', ingrese el nombre de la simulación y haga click en 'create'.

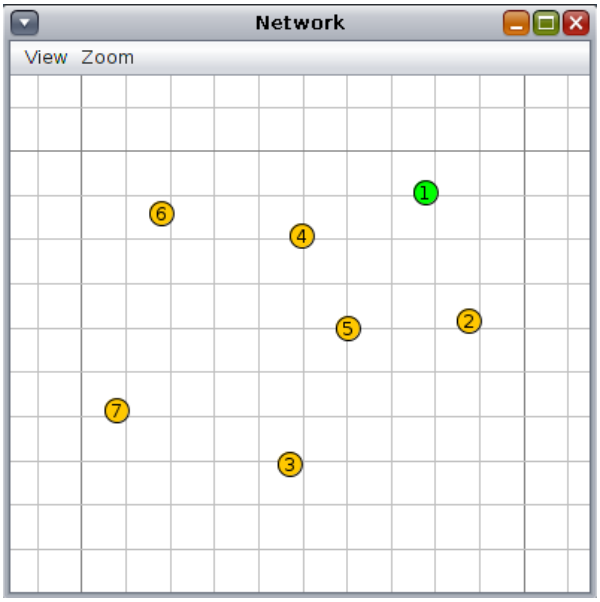
2. Haga click en 'Motes'. Del menú desplegable seleccione 'Add New Motes' seguido por 'Create new motes' y seleccione como Description “Router” o una descripción acorde a la función y como *mote type* 'Cooja mote…'.

3. Como programa navegue a 'contiki-ng/examples/rpl-border-router' y seleccione el archivo rpl-border-router.c. Haga click en 'Create'. Agregue un *mote* de este tipo.

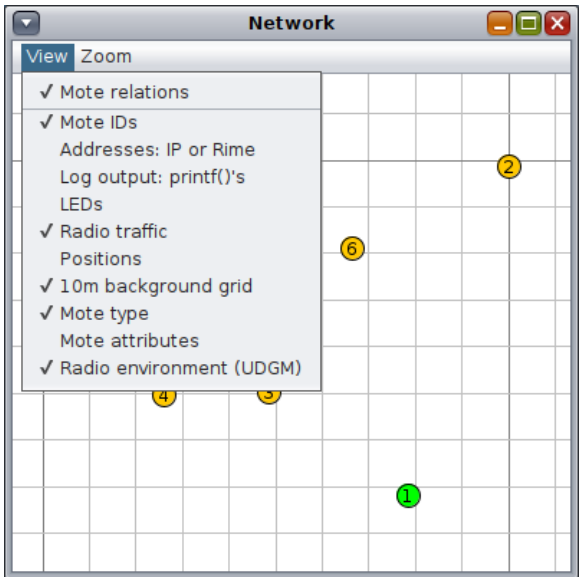
4. Repita los pasos 2 y 3 pero esta vez navegue a ‘contiki-ng/examples /hello-world' y seleccione el archivo hello-world.c. Agregue 6 - 7 motes del tipo hello-world.

5. Una vez que los motes hayan sido agregados se los puede reposicionar. Observe que si está activada la vista “Radio environment” es posible visualizar el alcance de los transmisores, asegúrese que se pueda formar una red y que ésta sea lo suficientemente compleja como para que se observe la formación de las rutas.

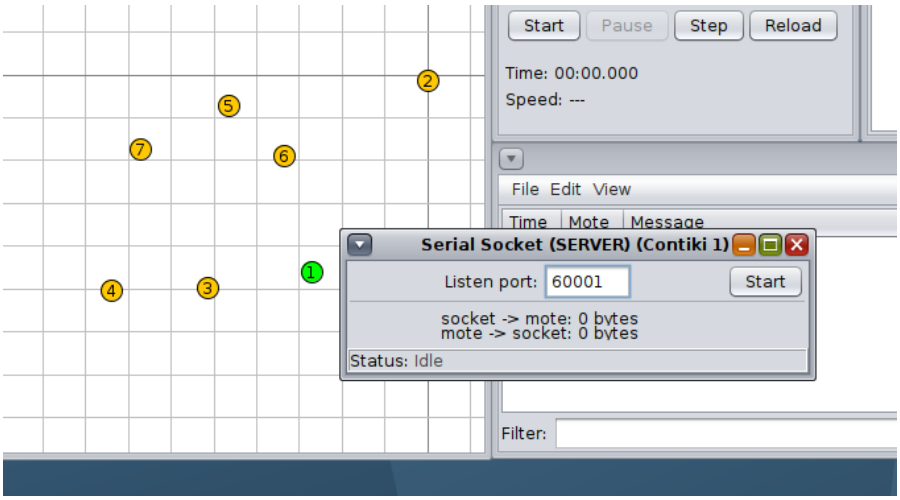
6. En la ventana de “Radio Messages” habilite la opción de 6LoWPAN Analyzer con PCAP.



Seleccione las opciones en **View** como se muestra a continuación. Esto lo ayudará a crear una topología de su elección.



6. Ahora, necesitamos crear un puente entre la red RPL simulada en Cooja y la máquina local. Esto se puede hacer haciendo clic derecho en el *mote* que está programado como *border router*. Seleccione 'More tools for border router' y luego 'Serial Socket (SERVER)'. Le va a aparecer el siguiente mensaje tras la ejecución exitosa de este paso. Note que el mensaje diga 'Listening on port 60001'



7. Arranque la simulación haciendo click en Start

Tunslip utility

Como se mencionó en la introducción, un enrutador de borde ayuda a conectar una red a otra. En este ejemplo, el enrutador de borde se utiliza para enrutar datos entre una red RPL y una red externa. Hasta ahora solo hemos creado la red RPL. Ahora necesitamos simular el escenario en el que esta red RPL está conectada a una red externa. Para este propósito usaremos la utilidad *tunslip* provista en Contiki. En este ejemplo, tunslip crea un puente entre la red RPL y la máquina local. tunslip6.c se puede encontrar en contiki-ng/tolos/serial-io . Compile el código tunslip6.

```
make tunslip6
```

(`tunslip` usa `ifconfig` que ha sido reemplazado por el comando `ip` en las nuevas distribuciones de Linux. Si llega a tener un error por este motivo, instale el paquete *net-utils* con `apt install net-utils`)

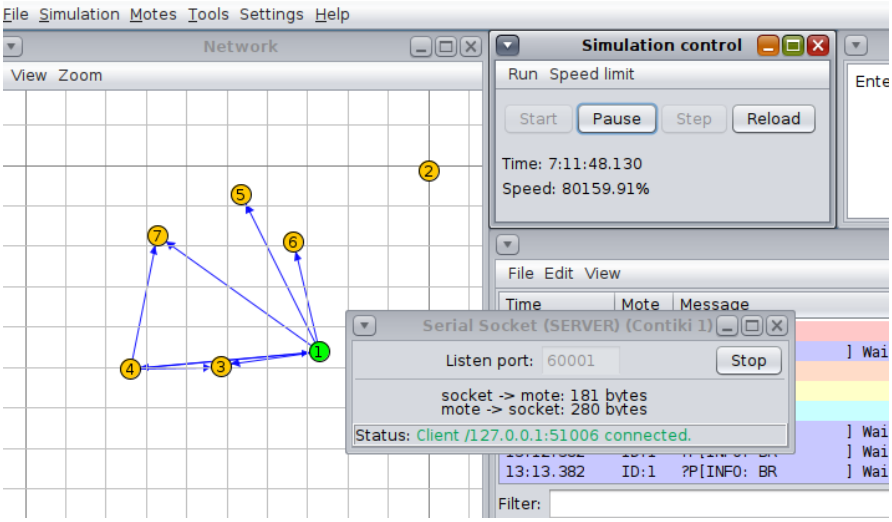
Haga una conexión entre su máquina local y el border router:

```
sudo ./tunslip6 -a 127.0.0.1 aaaa::1/64
```

Tras la ejecución exitosa del comando aparecerá la siguiente información en la terminal:

```
[sudo] password for user:
slip connected to ``127.0.0.1:60001''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` mtu 1500 up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 127.0.1.1  netmask 255.255.255.255  destination 127.0.1.1
        inet6 aaaa::1  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::c29d:f247:b207:debb  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::1  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

 [INFO: BR        ] Waiting for prefix
*** Address:aaaa::1 => aaaa:0000:0000:0000

. . .

[INFO: BR        ] Waiting for prefix
*** Address:aaaa::1 => aaaa:0000:0000:0000
[INFO: BR        ] Waiting for prefix
[INFO: BR        ] Server IPv6 addresses:
[INFO: BR        ]   aaaa::201:1:1:1
[INFO: BR        ]   fe80::201:1:1:1
```

Regrese al GUI del simulador Cooja y observe el cuadro de diálogo del servicio. El mensaje ahora ha cambiado a 'Client connected: /127.0.0.1'.



Verificando los resultados

Se puede verificar que se haya establecido la dirección del border router usando el comando ping.

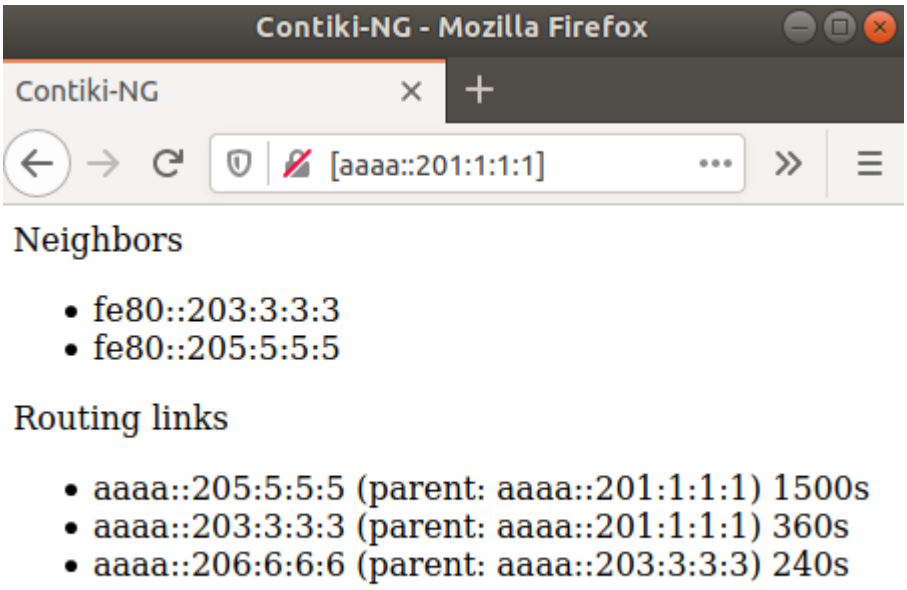
```
user@contiki-ng:~$ ping aaaa::201:1:1:1
PING aaaa::201:1:1:1(aaaa::201:1:1:1) 56 data bytes
64 bytes from aaaa::201:1:1:1: icmp_seq=1 ttl=64 time=21.6 ms
64 bytes from aaaa::201:1:1:1: icmp_seq=2 ttl=64 time=5.52 ms
64 bytes from aaaa::201:1:1:1: icmp_seq=3 ttl=64 time=3.98 ms
64 bytes from aaaa::201:1:1:1: icmp_seq=4 ttl=64 time=3.90 ms
64 bytes from aaaa::201:1:1:1: icmp_seq=5 ttl=64 time=14.6 ms
64 bytes from aaaa::201:1:1:1: icmp_seq=6 ttl=64 time=357 ms
^C
--- aaaa::201:1:1:1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5013ms
rtt min/avg/max/mdev = 3.900/67.902/357.770/129.793 ms
```

Se puede hacer ping a cualquier nodo de la red. Aquí hacemos ping al nodo 6

```
user@contiki-ng:~$ ping aaaa::206:6:6:6
PING aaaa::206:6:6:6(aaaa::206:6:6:6) 56 data bytes
64 bytes from aaaa::206:6:6:6: icmp_seq=1 ttl=62 time=128 ms
64 bytes from aaaa::206:6:6:6: icmp_seq=2 ttl=62 time=77.6 ms
64 bytes from aaaa::206:6:6:6: icmp_seq=3 ttl=62 time=65.2 ms
64 bytes from aaaa::206:6:6:6: icmp_seq=4 ttl=62 time=287 ms
64 bytes from aaaa::206:6:6:6: icmp_seq=5 ttl=62 time=76.2 ms
^C
--- aaaa::206:6:6:6 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4070ms
rtt min/avg/max/mdev = 65.278/127.005/287.325/83.099 ms
```

La dirección IPv6 de un nodo se puede verificar seleccionando “Addresses” en la ventana “Network”.

Se puede ingresar la dirección del border router en un web browser, se mostrará una página con los vecinos y rutas como se muestra a continuación:



Los mensajes pueden analizarse con Wireshark si se habilitó la opción Analyzer -> 6LoWPAN analyzer with PCAP en la ventana “Radio Messages”. El archivo generado está en ~/contiki-ng/tools/cooja/build y el nombre es radiolog-XXXX.pcap donde XXXX es un número secuencial. Para identificar el archivo una opción es ordenarlos por fecha de creación decreciente.

Problemas comunes

1. Asegúrese de ingresar correctamente la dirección IPv6 en el comando sudo ./tunslip6 -a 127.0.0.1 aaaa::1/64. No debe haber espacios en aaaa::1/64. De otro modo el código puede tener un comportamiento impredecible.

2. Cuando se pausa la simulación la conexión SLIP se pierde y podría no funcionar correctamente si se resume la simulación. En esos casos recargue la simulación y cree una nueva conexión SLIP.

Preguntas

1. Identifique los mensajes de RPL. ¿cuántos tipos de mensaje puede diferenciar? Enumérelos.

2. ¿Mediante qué protocolo de IPv6 se establece el DODAG?

3. ¿Cómo lo identifica?

4. ¿Qué valor tiene el DODAGID en los mensajes RPL? ¿Con qué lo relaciona?