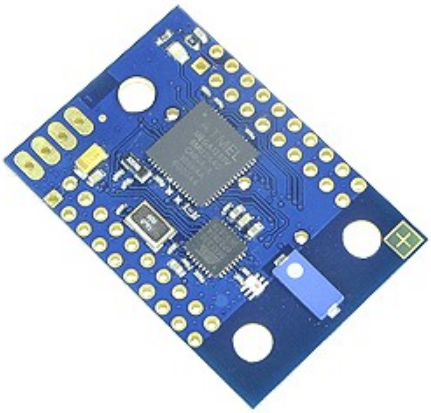
Atmel ATmega 1281 & Contiki

New Platform: avr-icm230_12

A new Contiki platform was created starting from the AVR-Zigbit platform and the AVR-Raven platform. A new folder was created: *~/Contiki/platform/avr-icm230_12/*

Some changes were also made in the *~/Contiki/cpu/avr/radio/rf230bb* files to use the radiotransmitter of the Atmel Atmega 1281 Modules and USB Sticks:



The ICradio Stick 2.4G is a compact USB-dongle, specified for ZigBee / IEEE 802.15.4 network applications. It is based on the ATmega128 controller and the AT86RF230 2.4GHz radio chip from Atmel.

The ICradio Module 2.4G is a compact and flexible to use radio, specified for ZigBee / IEEE 802.15.4 network applications. It's excellent for evaluation purposes, since most of the IO-pins of the ATmega128 are accessible on pinheads. It is based on the powerful ATmega128 controller and the AT86RF230 2.4GHz radio chip from Atmel.

Link to the ICradio Stick Datasheet

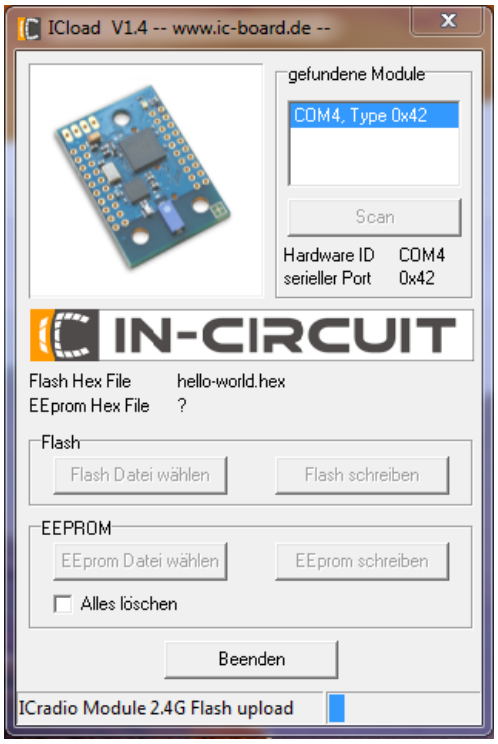
Link to the ICradio Module Datasheet

Hello World example

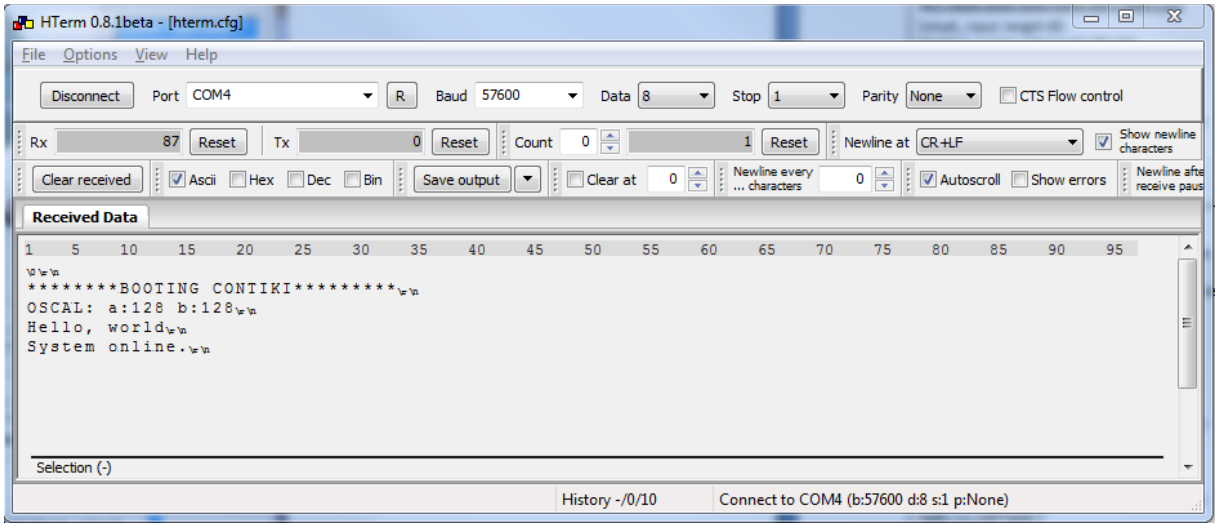
Hello world is the first example that has to be tried. First, the source code has to be compiled for the new platform that was created:

```
$ cd examples/hello world
$ make TARGET=avr-icm230_12 hello-world.hex
```

The .hex output file is then USB-flashed on the module with a software, *ICload*



With a software named *HTerm* it is possible to read the serial line output after setting up some parameters like the COM port and the baud rate (in this case the baud rate was chosen to 57600).



IPv6 Udp Sender & Receiver

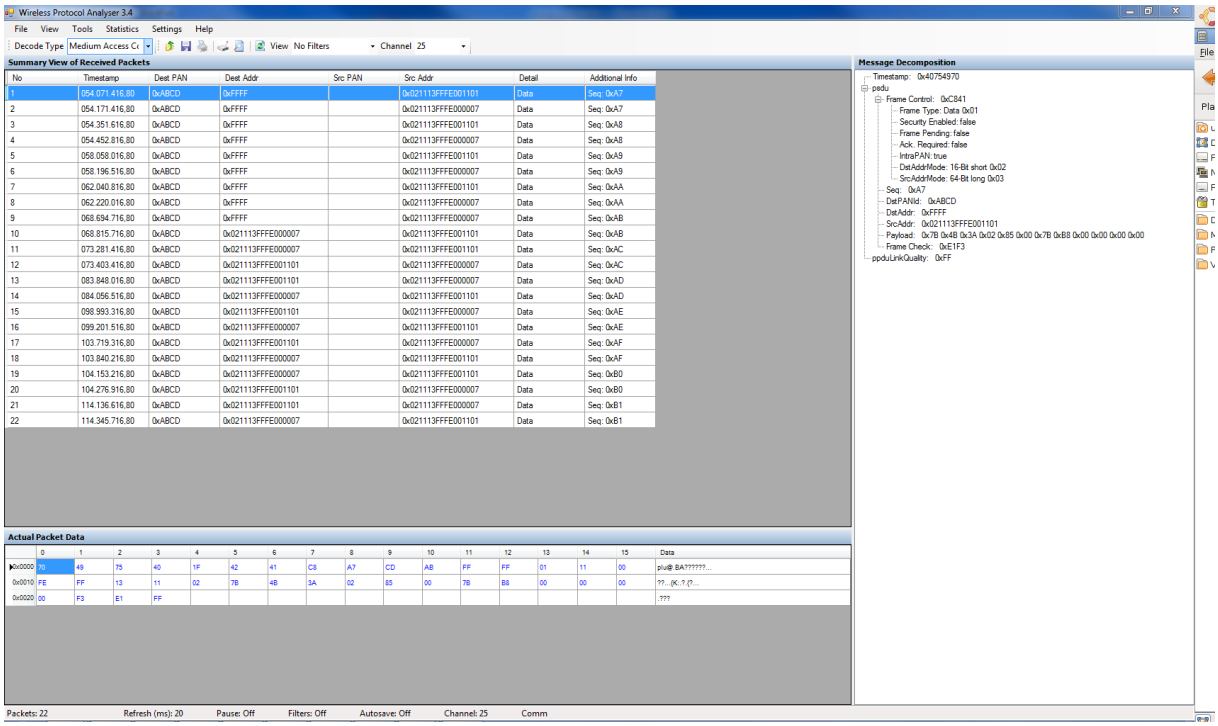
With this example it is possible to send from the udp-client module a string to the udp-server and to get an answer back when the packet is successfully received. Note that in the source code it is necessary to set the client/server IPv6 address and communication port.

```
$ cd examples/udp-ipv6/
$ make TARGET=avr-icm230_12 udp-server.hex udp-client.hex
```

From the serial line output it is possible to read the messages exchange:

UDP Client	*UDP Server*
********BOOTING CONTIKI*********	********BOOTING CONTIKI*********
UDP client process started	UDP server started
Client IPv6 addresses: fe80::11:13ff:fe00:7	Server IPv6 addresses: fe80::11:13ff:fe00:1101
Created a connection with the server	System online.
fe80::11:13ff:fe00:1101 local/remote port 3001/3000	
System online.	rf230_read: 29 bytes lqi 255 crc 1
rf230_read: 29 bytes lqi 255 crc 1	icmp6_input: length 48
icmp6_input: length 48	
	rf230_read: 66 bytes lqi 255 crc 1
rf230_read: 66 bytes lqi 255 crc 1	**IPv6 packet received from**
IPv6 packet received from	**fe80:0000:0000:0000:0011:13ff:fe00:0007 to**
fe80:0000:0000:0000:0011:13ff:fe00:1101 to	**ff02:0000:0000:0000:0000:0001:ff00:1101**
fe80:0000:0000:0000:0011:13ff:fe00:0007	icmp6_input: length 80
icmp6_input: length 80	Sending packet with length 80 (40)
rf230_read: 66 bytes lqi 255 crc 1	rf230_read: 66 bytes lqi 255 crc 1
IPv6 packet received from	IPv6 packet received from
fe80:0000:0000:0000:0011:13ff:fe00:1101 to	fe80:0000:0000:0000:0011:13ff:fe00:0007 to
fe80:0000:0000:0000:0011:13ff:fe00:0007	fe80:0000:0000:0000:0011:13ff:fe00:1101
icmp6_input: length 80	icmp6_input: length 80
Sending packet with length 80 (40)	
Client sending to: fe80::11:13ff:fe00:1101 (msg: Hello 1 from the client)	Receiving UDP packet
In udp_send	In udp_found
Sending packet with length 71 (31)	**Server received: 'Hello 1 from the client' from fe80::11:13ff:fe00:7**
	Responding with message: Hello from the server! (1)
rf230_read: 58 bytes lqi 255 crc 1	In udp_send
IPv6 packet received from	Sending packet with length 74 (34)
fe80:0000:0000:0000:0011:13ff:fe00:1101 to	In udp_send
fe80:0000:0000:0000:0011:13ff:fe00:0007	
Receiving UDP packet	rf230_read: 55 bytes lqi 255 crc 1
In udp_found	IPv6 packet received from
Response from the server: 'Hello from the server! (1)'	fe80:0000:0000:0000:0011:13ff:fe00:0007 to
In udp_send	fe80:0000:0000:0000:0011:13ff:fe00:1101
Client sending to: fe80::11:13ff:fe00:1101 (msg: Hello 2 from the client)	Receiving UDP packet
	In udp_found
	Server received: 'Hello 2 from the client' from fe80::11:13ff:fe00:7
	Responding with message: Hello from the server! (2)

With a 802.15.4 USB Stick and a software called *Wireless Protocol Analyzer* it is possible to sniff the packets in a specific channel (in this case, channel 25 was used)


Wireless Protocol Analyser 3.4
File View Tools Statistics Settings Help
Decode Type Medium Access Cc
View No Filters
Channel 25
Summary View of Received Packets
No Timestamp Dest PAN Dest Addr Src PAN Src Addr Detail Additional Info
1 054.071.416.80 0xABCD 0xFFFF 0x021113FFFE001101 Data Seq: 0xA7
2 054.171.416.80 0xABCD 0xFFFF 0x021113FFFE000007 Data Seq: 0xA7
3 054.351.616.80 0xABCD 0xFFFF 0x021113FFFE001101 Data Seq: 0xA8
4 054.452.816.80 0xABCD 0xFFFF 0x021113FFFE000007 Data Seq: 0xA8
5 058.058.016.80 0xABCD 0xFFFF 0x021113FFFE001101 Data Seq: 0xA9
6 058.196.516.80 0xABCD 0xFFFF 0x021113FFFE000007 Data Seq: 0xA9
7 062.040.816.80 0xABCD 0xFFFF 0x021113FFFE001101 Data Seq: 0xAA
8 062.220.016.80 0xABCD 0xFFFF 0x021113FFFE000007 Data Seq: 0xAA
9 068.694.716.80 0xABCD 0xFFFF 0x021113FFFE000007 Data Seq: 0xAB
10 068.815.716.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xAB
11 073.281.416.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xAC
12 073.403.416.80 0xABCD 0x021113FFFE001101 0x021113FFFE000007 Data Seq: 0xAC
13 083.848.016.80 0xABCD 0x021113FFFE001101 0x021113FFFE000007 Data Seq: 0xAD
14 084.056.516.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xAD
15 098.993.316.80 0xABCD 0x021113FFFE001101 0x021113FFFE000007 Data Seq: 0xAE
16 099.201.516.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xAE
17 103.719.316.80 0xABCD 0x021113FFFE001101 0x021113FFFE000007 Data Seq: 0xAF
18 103.840.216.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xAF
19 104.153.216.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xB0
20 104.276.916.80 0xABCD 0x021113FFFE001101 0x021113FFFE000007 Data Seq: 0xB0
21 114.136.616.80 0xABCD 0x021113FFFE001101 0x021113FFFE000007 Data Seq: 0xB1
22 114.345.716.80 0xABCD 0x021113FFFE000007 0x021113FFFE001101 Data Seq: 0xB1
Message Decomposition
Timestamp: 0x40754970
pedu
Frame Control: 0xC841
Frame Type: Data 0x01
Security Enabled: false
Frame Pending: false
Ack. Required: false
IntraPAN: true
DstAddrMode: 16-Bit short 0x02
SrcAddrMode: 64-Bit long 0x03
Seq: 0xA7
DstPANId: 0xABCD
DstAddr: 0xFFFF
SrcAddr: 0x021113FFFE001101
Payload: 0x7B 0x4B 0x3A 0x02 0x85 0x00 0x7B 0xB8 0x00 0x00 0x00 0x00
Frame Check: 0xE1F3
ppduLinkQuality: 0xFF
Actual Packet Data
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 Data
0x0000 70 49 75 40 1F 42 41 C8 A7 CD AB FF FF 01 11 00 plu@ BA??????...
0x0010 FE FF 13 11 02 7B 4B 3A 02 85 00 7B B8 00 00 00 ??...{K:.?.{?...
0x0020 00 F3 E1 FF ???
Packets: 22
Refresh (ms): 20
Pause: Off
Filters: Off
Autosave: Off
Channel: 25
Comm

RPL Collect: Compiling Sink node & Sender node

```
$ cd examples/ipv6/rpl-collect
$ make TARGET=avr-icm230_12 udp-sink.hex udp-sender.hex
```

The RPL Debug Messages are useful to understand the topology of the network and it is possible to enable them in the file *~/Contiki/core/net/rpl/rpl-icmp6.c*
If only the neighbor informations are needed, it is possible to enable the Debug messages in the file *~/Contiki/core/net/neighbor-info.c*

```
//#define DEBUG DEBUG_NONE
#define DEBUG DEBUG_FULL
```

With the following command it is possible to check the memory usage of the hex file:

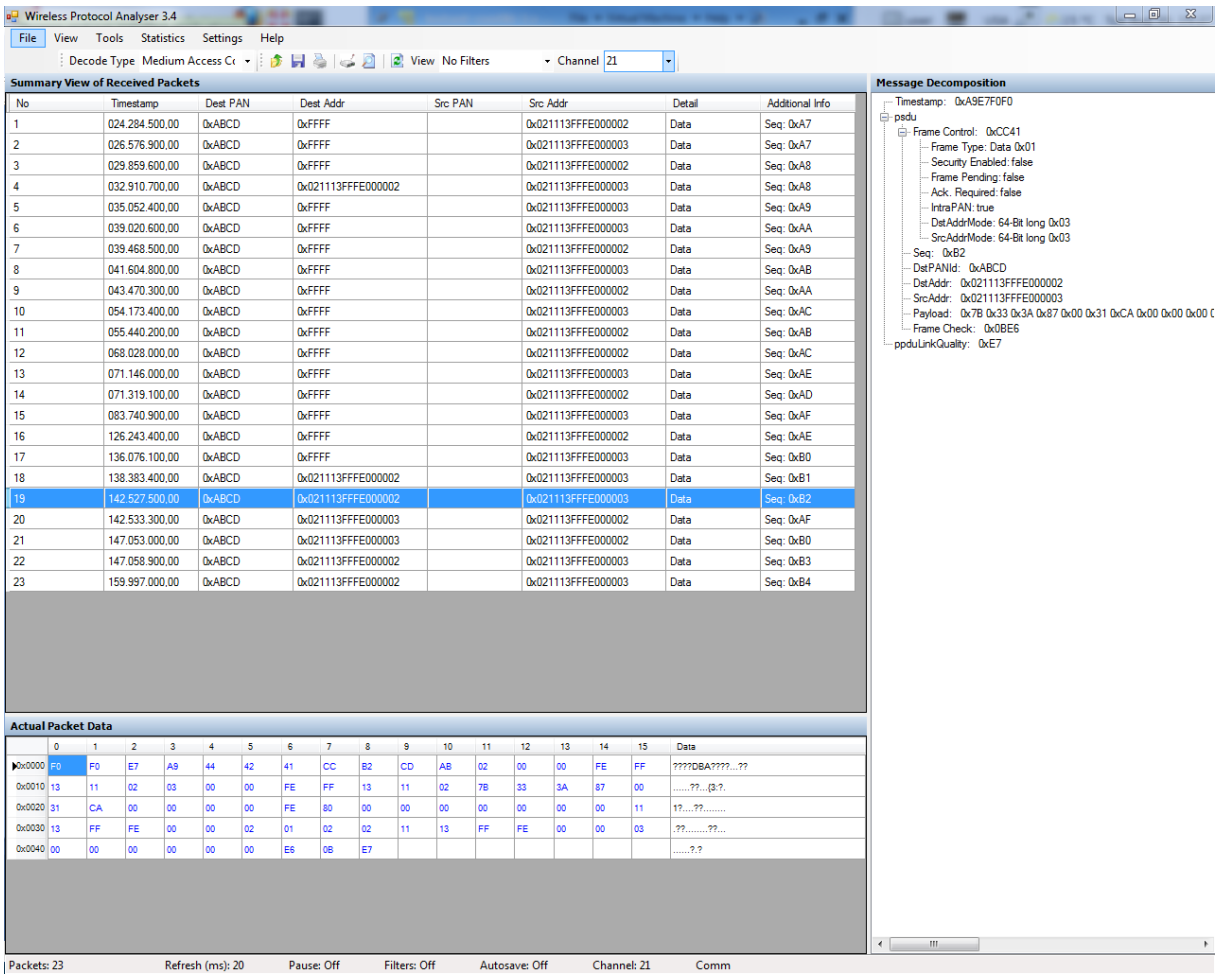
```
$ avr-size -C udp-sink.avr-icm230_12
AVR Memory Usage
----------------
Device: Unknown

Program:   51448 bytes
(.text + .data + .bootloader)

Data:       4476 bytes
(.data + .bss + .noinit)

EEPROM:        8 bytes
(.eeprom)
```

If the sender node is going to be a leaf node, it is possible to enable a flag to change the RPL ranking and force it to have always infinite rank in order to reduce the power consumption. The flag which has to be enabled is in *~/Contiki/platform/avr-icm230_12/contiki-conf.h*

```
RPL_CONF_LEAF_NODE    1
```

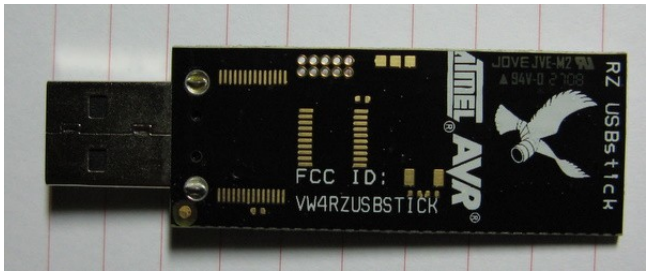
The serial line output is useful to see how the RPL is working:

UDP-Sender	*UDP-Sink*
********BOOTING CONTIKI********* OSCAL: a:130 b:130 System online. UDP client process started Client IPv6 addresses: bbbb::11:13ff:fe00:3 fe80::11:13ff:fe00:3 Created a connection with the server :: local/remote port 8775/5688	********BOOTING CONTIKI********* OSCAL: a:128 b:128 I am sink! System online. UDP server started created a new RPL dag Server IPv6 addresses: :: bbbb::2 fe80::11:13ff:fe00:2 Created a server connection with remote address :: local/remote port 5688/8775
RPL: Sending a DIS Received an RPL control message RPL: Received a DIO from fe80::11:13ff:fe00:2 RPL: Neighbor added to neighbor cache fe80::11:13ff:fe00:2, 02:11:13:ff:fe:00:00:02 RPL: Incoming DIO rank 256 RPL: DIO suboption 2, length: 6 RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 0 RPL: DIO suboption 4, length: 14 RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535 RPL: DIO suboption 8, length: 30 RPL: Copying prefix information	 RPL: Sending prefix info in DIO for bbbb:: RPL: Sending a multicast-DIO with rank 256 Received an RPL control message RPL: Received a DIO from fe80::11:13ff:fe00:3 RPL: Neighbor added to neighbor cache fe80::11:13ff:fe00:7, 02:11:13:ff:fe:00:00:03
RPL: Sending prefix info in DIO for bbbb:: RPL: Sending a multicast-DIO with rank 1536 Received an RPL control message RPL: Received a DIO from fe80::11:13ff:fe00:2 RPL: Neighbor already in neighbor cache RPL: Incoming DIO rank 256 RPL: DIO suboption 2, length: 6 RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 0 RPL: DIO suboption 4, length: 14 RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535 RPL: DIO suboption 8, length: 30 RPL: Copying prefix information RPL: Sending DAO with prefix bbbb::11:13ff:fe00:3 to fe80::11:13ff:fe00:2	RPL: Incoming DIO rank 1536 RPL: DIO suboption 2, length: 6 RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 640 RPL: DIO suboption 4, length: 14 RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535 RPL: DIO suboption 8, length: 30 RPL: Copying prefix information RPL: Sending prefix info in DIO for bbbb:: RPL: Sending a multicast-DIO with rank 256 Received an RPL control message RPL: Received a DAO from fe80::11:13ff:fe00:3 RPL: DAO lifetime: 255, prefix length: 128 prefix: bbbb::11:13ff:fe00:3
	30 0 120 0 7 1 1 0 22 14616 0 0 0 0 0 512 8 512 1 131 0 0 0 0 0 0 0 0 0 0 0 30 0 141 0 7 2 1 0 22 17305 0 0 0 0 0 512 8 512 1 131 0 0 0 0 0 0 0 0 0 0 0


Wireless Protocol Analyser 3.4
File View Tools Statistics Settings Help
Decode Type Medium Access Cc
View No Filters
Channel 21
Summary View of Received Packets
No Timestamp Dest PAN Dest Addr Src PAN Src Addr Detail Additional Info
1 024.284.500,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xA7
2 026.576.900,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xA7
3 029.859.600,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xA8
4 032.910.700,00 0xABCD 0x021113FFFE000002 0x021113FFFE000003 Data Seq: 0xA8
5 035.052.400,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xA9
6 039.020.600,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xAA
7 039.468.500,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xA9
8 041.604.800,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xAB
9 043.470.300,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xAA
10 054.173.400,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xAC
11 055.440.200,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xAB
12 068.028.000,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xAC
13 071.146.000,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xAE
14 071.319.100,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xAD
15 083.740.900,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xAF
16 126.243.400,00 0xABCD 0xFFFF 0x021113FFFE000002 Data Seq: 0xAE
17 136.076.100,00 0xABCD 0xFFFF 0x021113FFFE000003 Data Seq: 0xB0
18 138.383.400,00 0xABCD 0x021113FFFE000002 0x021113FFFE000003 Data Seq: 0xB1
19 142.527.500,00 0xABCD 0x021113FFFE000002 0x021113FFFE000003 Data Seq: 0xB2
20 142.533.300,00 0xABCD 0x021113FFFE000003 0x021113FFFE000002 Data Seq: 0xAF
21 147.053.000,00 0xABCD 0x021113FFFE000003 0x021113FFFE000002 Data Seq: 0xB0
22 147.058.900,00 0xABCD 0x021113FFFE000002 0x021113FFFE000003 Data Seq: 0xB3
23 159.997.000,00 0xABCD 0x021113FFFE000002 0x021113FFFE000003 Data Seq: 0xB4
Actual Packet Data
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 Data
0x0000 F0 F0 E7 A9 44 42 41 CC B2 CD AB 02 00 00 FE FF ????DBA????..??
0x0010 13 11 02 03 00 00 FE FF 13 11 02 7B 33 3A 87 00 .....??...(3:?.
0x0020 31 CA 00 00 00 00 FE 80 00 00 00 00 00 00 00 11 1?....??.....
0x0030 13 FF FE 00 00 02 01 02 02 11 13 FF FE 00 00 03 .??.....??...
0x0040 00 00 00 00 00 00 E6 0B E7 ..... ?.?
Message Decomposition
Timestamp: 0xA9E7F0F0
psdu
Frame Control: 0xCC41
Frame Type: Data 0x01
Security Enabled: false
Frame Pending: false
Ack. Required: false
IntraPAN: true
DstAddrMode: 64-Bit long 0x03
SrcAddrMode: 64-Bit long 0x03
Seq: 0xB2
DstPANId: 0xABCD
DstAddr: 0x021113FFFE000002
SrcAddr: 0x021113FFFE000003
Payload: 0x7B 0x33 0x3A 0x87 0x00 0x31 0xCA 0x00 0x00 0x00
Frame Check: 0x0BE6
ppduLinkQuality: 0xE7
Packets: 23
Refresh (ms): 20
Pause: Off
Filters: Off
Autosave: Off
Channel: 21
Comm

RPL Border Router

An AVR-Raven USB Stick (Jackdaw) was used as a Border Router, following the Contiki-Wiki configuration tutorial.



The Jackdaw is normally a repeater with minimal ipv6 routines implemented via fakeuip.c, but RPL can be added by including the full uip6 stack in the makefile
~/Contiki/examples/ravenusbstick/Makefile.ravenusbstick

```
#CONTIKI_NO_NET=1
UIP_CONF_IPV6=1
```

The firmware has to be compiled and flashed in the Jackdaw Stick with a linux application named dfu-programmer:

```
$ cd examples/ravensubstik
$ make
$ sudo dfu-programmer at90usb1287 erase
$ sudo dfu-programmer at90usb1287 flash ravenusbstick.hex
$ sudo dfu-programmer at90usb1287 start
```

In order to turn on the Border router, a make start command is necessary:

```
$ cd examples/ravenusbstick/
$ sudo dfu-programmer at90usb1287 start
```

By default RNDIS does the ipv6 address resolution protocol of neighbor solicitation/neighbor advertisement, and RPL doesn't support link-layer NS/NA broadcasts (even if it did you would not want the extra traffic). Without a NA response RNDIS will not send any IPV6 addressed packets.

Although the Jackdaw could trap the NS and construct a NA reply, it is simple to disable NS/NA on the Linux RNDIS interface:

```
$ ifconfig usb0 -arp
$ ip -6 address add bbbb::0/64 dev usb0
```

Or, alternatively,

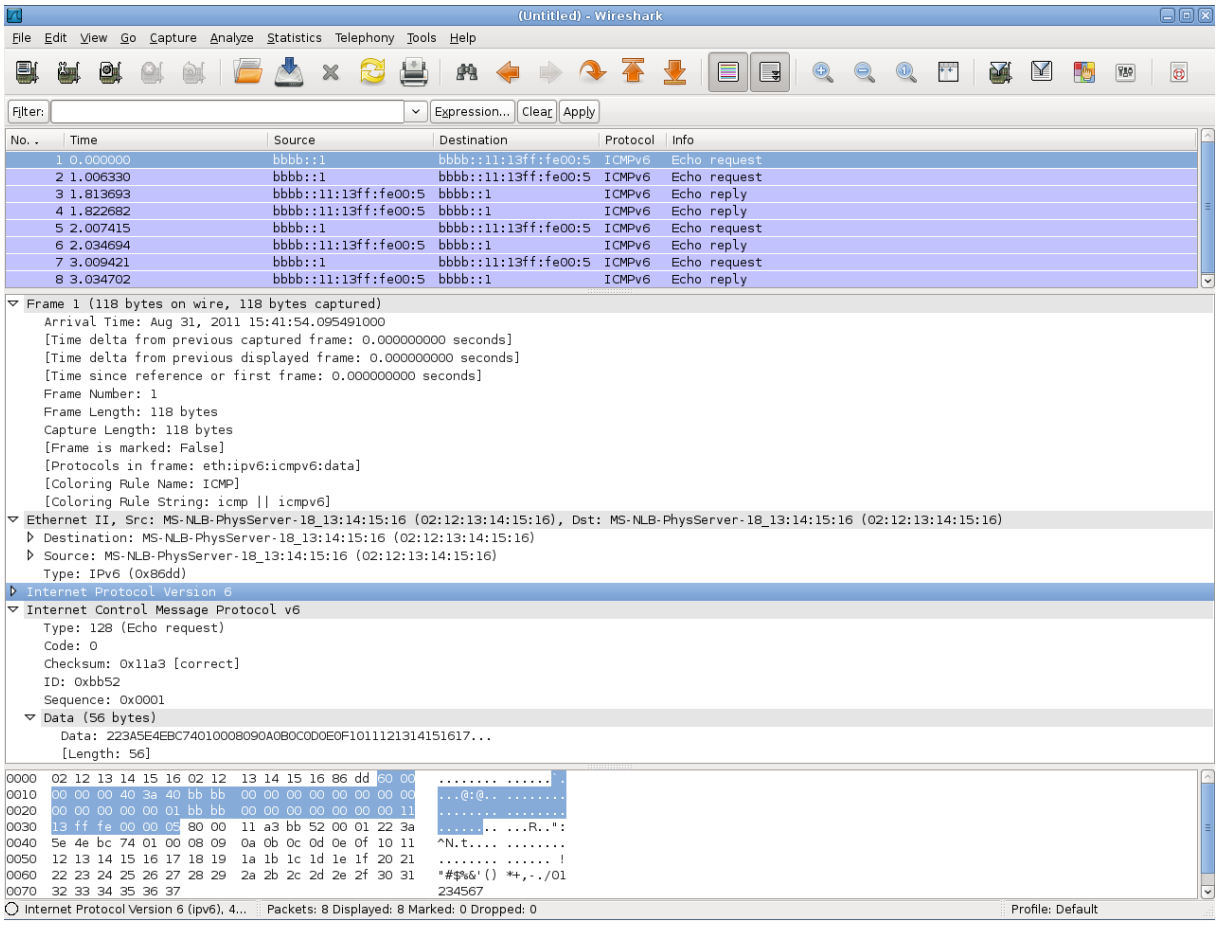
```
$ make usb0up
```

Once the Border Router is turned on, it is possible to ping the interface at the address bbbb::1 and the border router at bbbb::200 and every neighbor node at its IPv6 address

```
$ ping6 bbbb::1
```

```
PING bbbb::1(bbbb::1) 56 data bytes
64 bytes from bbbb::1: icmp_seq=1 ttl=64 time=0.103 ms
64 bytes from bbbb::1: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from bbbb::1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from bbbb::1: icmp_seq=4 ttl=64 time=0.030 ms

$ ping6 bbbb::200
PING bbbb::200(bbbb::200) 56 data bytes
64 bytes from bbbb::200: icmp_seq=1 ttl=64 time=4.11 ms
64 bytes from bbbb::200: icmp_seq=2 ttl=64 time=10.2 ms
64 bytes from bbbb::200: icmp_seq=3 ttl=64 time=10.0 ms
64 bytes from bbbb::200: icmp_seq=4 ttl=64 time=6.93 ms

$ ping6 bbbb::11:13ff:fe00:5
PING bbbb::11:13ff:fe00:5(bbbb::11:13ff:fe00:5) 56 data bytes
64 bytes from bbbb::11:13ff:fe00:5: icmp_seq=2 ttl=63 time=30.7 ms
64 bytes from bbbb::11:13ff:fe00:5: icmp_seq=3 ttl=63 time=27.5 ms
64 bytes from bbbb::11:13ff:fe00:5: icmp_seq=4 ttl=63 time=25.5 ms
64 bytes from bbbb::11:13ff:fe00:5: icmp_seq=5 ttl=63 time=23.5 ms
```

With Wireshark it is possible to analyze the packets during the ping session:



Through the debug console output it is possible to get some information about the Jackdaw Border router and the neighbors available. First of all it's necessary to connect with Putty or with CU with the following command:

```
$ cu -l /dev/ttyACM0 --nostop -s57600
```

By pressing the key *h* or the key *?* it is possible to view the Jackdaw RPL Border Menu

```
********** Jackdaw Menu **********
        [Built Aug 31 2011]
*  m        Print current mode    *
*  s        Set to sniffer mode   *
*  n        Set to network mode   *
*  c        Set RF channel        *
*  p        Set RF power          *
*  6        Toggle 6lowpan        *
*  r        Toggle raw mode       *
*  d        Toggle RS232 output   *
*  S        Enable sneezer mode   *
*  N        RPL Neighbors         *
*  G        RPL Global Repair     *
*  e        Energy Scan           *
*  D        Switch to DFU mode    *
*  R        Reset (via WDT)       *
*  h,?      Print this menu       *
*                                 *
*   Fraunhofher Fokus Institute   *
***********************************
```

The key *m* gives the possibility to see some useful information about the Jackdaw Border Router:

```
Currently Jackdaw:
  * Will send data over RF
  * Will change link-local addresses inside IP messages
  * Will decompress 6lowpan headers
  * Will not Output raw 802.15.4 frames
  * Will Output RS232 debug strings
  * USB Ethernet MAC: 02:12:13:14:15:16
  * 802.15.4 EUI-64: 02:12:13:ff:fe:14:15:16
  * Operates on channel 21 with TX power  +3.0dBm
  * Current/Last/Smallest RSSI: -88/-88/-91dBm
  * Configuration: 129, USB<->ETH is active
  * Never-used stack > 800 bytes
```

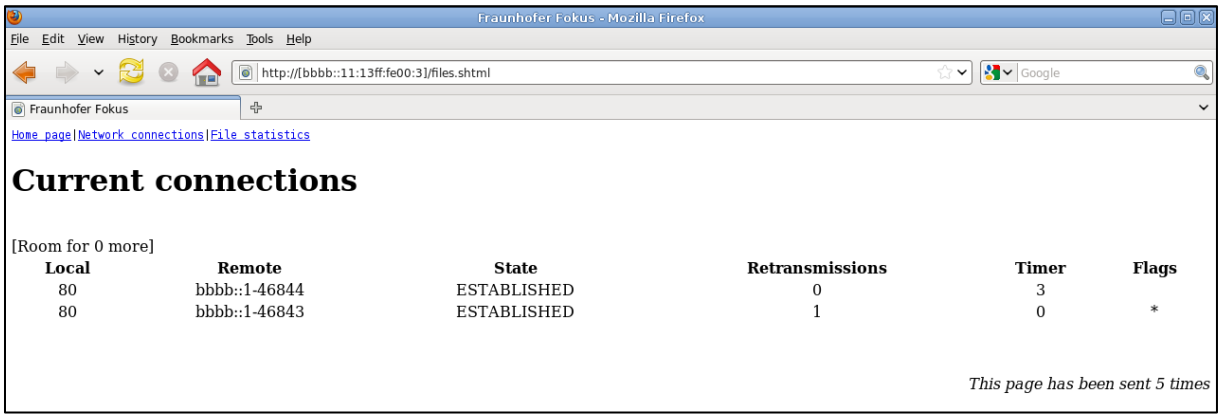
And the key N gives the possibility to see the neighbors and routes of the Border Router:

```
Addresses [4 max]
bbbb::200
fe80::12:13ff:fe14:1516
```

```
Neighbors [3 max]
fe80::11:13ff:fe00:9
fe80::11:13ff:fe00:3
fe80::11:13ff:fe00:5

Routes [3 max]
bbbb::11:13ff:fe00:3/128 (via fe80::11:13ff:fe00:3)
bbbb::11:13ff:fe00:9/128 (via fe80::11:13ff:fe00:5)
bbbb::11:13ff:fe00:5/128 (via fe80::11:13ff:fe00:5)
```

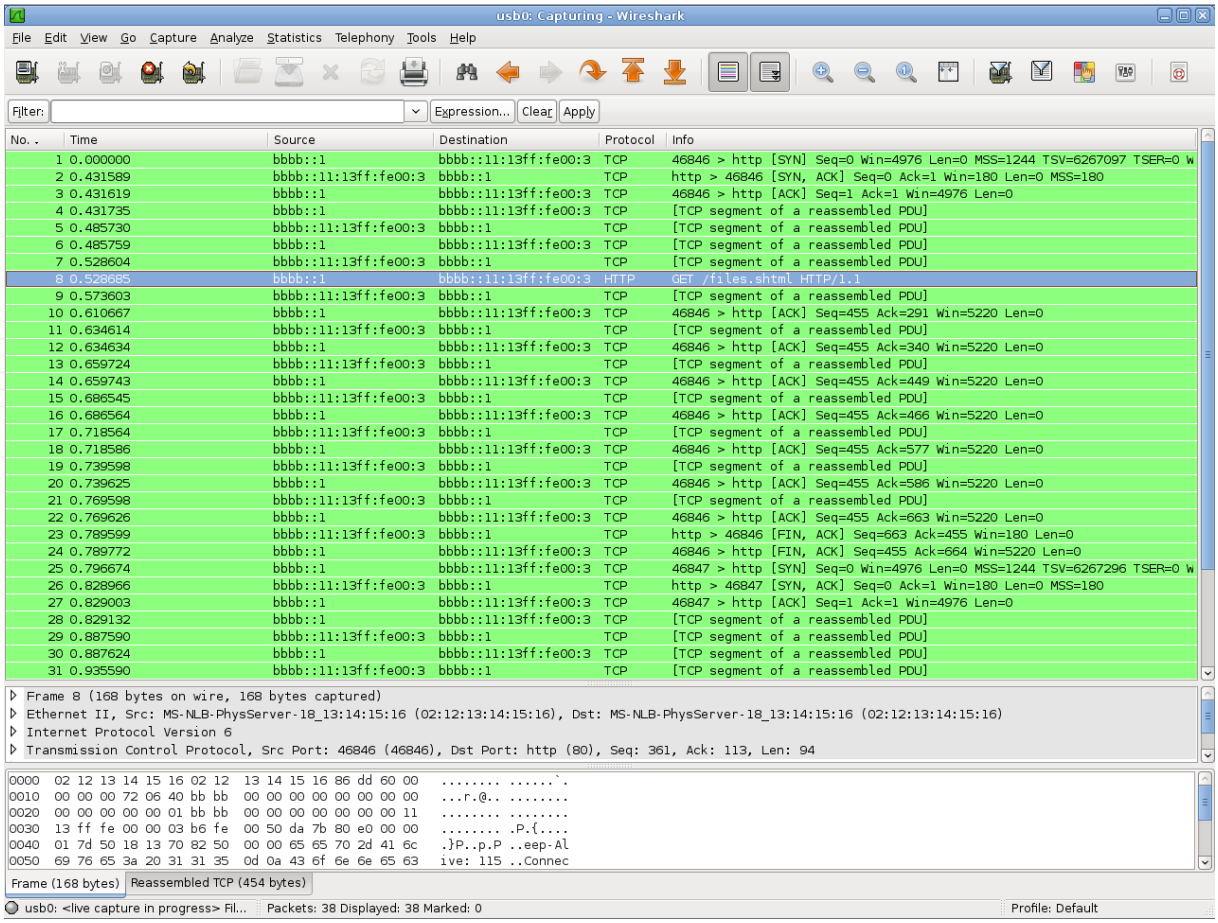
It is important to notice that it is possible to reach and ping the node 9 (`bbbb::11:13ff:fe00:9`) with a multihop through the node 5 (`bbbb::11:13ff:fe00:5`).

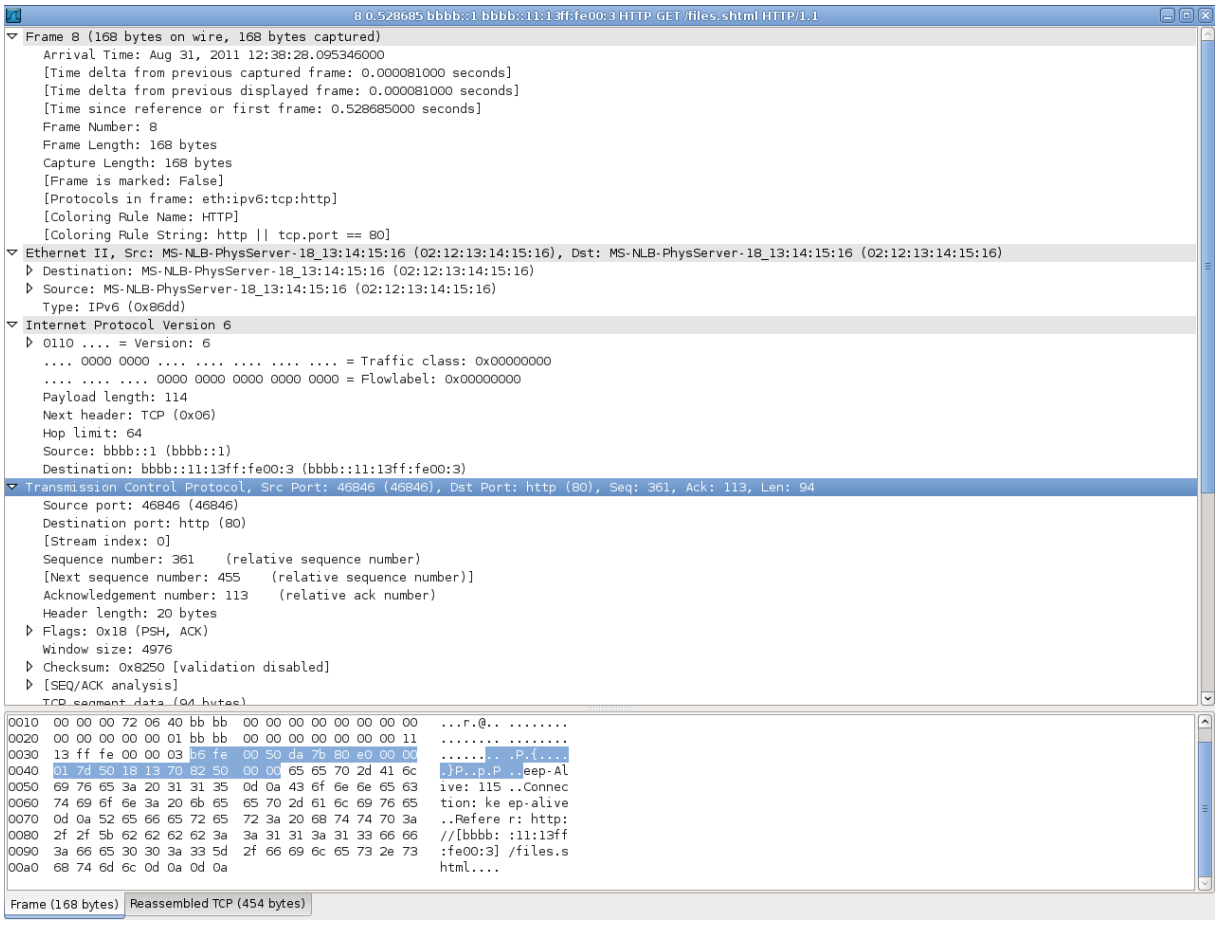
If one of the neighbors runs a webserver, the border router allows the access to it through its IPv6 address:



It is possible to check the packets that goes in and out the AVR Raven USB Stick with a very high level of protocol details with Wireshark on the usb0 interface :

```
$ sudo wireshark -i usb0
```


usb0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply
No. . Time Source Destination Protocol Info
1 0.000000 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [SYN] Seq=0 Win=4976 Len=0 MSS=1244 TSV=6267097 TSER=0 W
2 0.431589 bbbb::11:13ff:fe00:3 bbbb::1 TCP http > 46846 [SYN, ACK] Seq=0 Ack=1 Win=180 Len=0 MSS=180
3 0.431619 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=1 Ack=1 Win=4976 Len=0
4 0.431735 bbbb::1 bbbb::11:13ff:fe00:3 TCP [TCP segment of a reassembled PDU]
5 0.485730 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
6 0.485759 bbbb::1 bbbb::11:13ff:fe00:3 TCP [TCP segment of a reassembled PDU]
7 0.528604 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
8 0.528685 bbbb::1 bbbb::11:13ff:fe00:3 HTTP GET /files.shtml HTTP/1.1
9 0.573603 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
10 0.610667 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=291 Win=5220 Len=0
11 0.634614 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
12 0.634634 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=340 Win=5220 Len=0
13 0.659724 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
14 0.659743 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=449 Win=5220 Len=0
15 0.686545 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
16 0.686564 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=466 Win=5220 Len=0
17 0.718564 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
18 0.718586 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=577 Win=5220 Len=0
19 0.739598 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
20 0.739625 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=586 Win=5220 Len=0
21 0.769598 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
22 0.769626 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [ACK] Seq=455 Ack=663 Win=5220 Len=0
23 0.789599 bbbb::11:13ff:fe00:3 bbbb::1 TCP http > 46846 [FIN, ACK] Seq=663 Ack=455 Win=180 Len=0
24 0.789772 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46846 > http [FIN, ACK] Seq=455 Ack=664 Win=5220 Len=0
25 0.796674 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46847 > http [SYN] Seq=0 Win=4976 Len=0 MSS=1244 TSV=6267296 TSER=0 W
26 0.828966 bbbb::11:13ff:fe00:3 bbbb::1 TCP http > 46847 [SYN, ACK] Seq=0 Ack=1 Win=180 Len=0 MSS=180
27 0.829003 bbbb::1 bbbb::11:13ff:fe00:3 TCP 46847 > http [ACK] Seq=1 Ack=1 Win=4976 Len=0
28 0.829132 bbbb::1 bbbb::11:13ff:fe00:3 TCP [TCP segment of a reassembled PDU]
29 0.887590 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
30 0.887624 bbbb::1 bbbb::11:13ff:fe00:3 TCP [TCP segment of a reassembled PDU]
31 0.935590 bbbb::11:13ff:fe00:3 bbbb::1 TCP [TCP segment of a reassembled PDU]
Frame 8 (168 bytes on wire, 168 bytes captured)
Ethernet II, Src: MS-NLB-PhysServer-18_13:14:15:16 (02:12:13:14:15:16), Dst: MS-NLB-PhysServer-18_13:14:15:16 (02:12:13:14:15:16)
Internet Protocol Version 6
Transmission Control Protocol, Src Port: 46846 (46846), Dst Port: http (80), Seq: 361, Ack: 113, Len: 94
0000 02 12 13 14 15 16 02 12 13 14 15 16 86 dd 60 00 ....`.
0010 00 00 00 72 06 40 bb bb 00 00 00 00 00 00 00 00 ...r.@.. .....
0020 00 00 00 00 00 01 bb bb 00 00 00 00 00 00 00 11 ....
0030 13 ff fe 00 00 03 b6 fe 00 50 da 7b 80 e0 00 00 ..... .P.{....
0040 01 7d 50 18 13 70 82 50 00 00 65 65 70 2d 41 6c .}P..p.P ..eep-Al
0050 69 76 65 3a 20 31 31 35 0d 0a 43 6f 6e 6e 65 63 ive: 115 ..Connec
Frame (168 bytes) Reassembled TCP (454 bytes)
usb0: <live capture in progress> Fil... Packets: 38 Displayed: 38 Marked: 0 Profile: Default


8 0.528685 bbbb::1 bbbb::11:13ff:fe00:3 HTTP GET /files.shtml HTTP/1.1
Frame 8 (168 bytes on wire, 168 bytes captured)
Arrival Time: Aug 31, 2011 12:38:28.095346000
[Time delta from previous captured frame: 0.000081000 seconds]
[Time delta from previous displayed frame: 0.000081000 seconds]
[Time since reference or first frame: 0.528685000 seconds]
Frame Number: 8
Frame Length: 168 bytes
Capture Length: 168 bytes
[Frame is marked: False]
[Protocols in frame: eth:ipv6:tcp:http]
[Coloring Rule Name: HTTP]
[Coloring Rule String: http || tcp.port == 80]
Ethernet II, Src: MS-NLB-PhysServer-18_13:14:15:16 (02:12:13:14:15:16), Dst: MS-NLB-PhysServer-18_13:14:15:16 (02:12:13:14:15:16)
Destination: MS-NLB-PhysServer-18_13:14:15:16 (02:12:13:14:15:16)
Source: MS-NLB-PhysServer-18_13:14:15:16 (02:12:13:14:15:16)
Type: IPv6 (0x86dd)
Internet Protocol Version 6
0110 .... = Version: 6
.... 0000 0000 .... = Traffic class: 0x00000000
.... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 114
Next header: TCP (0x06)
Hop limit: 64
Source: bbbb::1 (bbbb::1)
Destination: bbbb::11:13ff:fe00:3 (bbbb::11:13ff:fe00:3)
Transmission Control Protocol, Src Port: 46846 (46846), Dst Port: http (80), Seq: 361, Ack: 113, Len: 94
Source port: 46846 (46846)
Destination port: http (80)
[Stream index: 0]
Sequence number: 361 (relative sequence number)
[Next sequence number: 455 (relative sequence number)]
Acknowledgement number: 113 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 4976
Checksum: 0x8250 [validation disabled]
[SEQ/ACK analysis]
TCP segment data (94 bytes)
0010 00 00 00 72 06 40 bb bb 00 00 00 00 00 00 00 00 ...r.@.. .....
0020 00 00 00 00 00 01 bb bb 00 00 00 00 00 00 00 11 ....
0030 13 ff fe 00 00 03 b6 fe 00 50 da 7b 80 e0 00 00 .....P.{....
0040 01 7d 50 18 13 70 82 50 00 00 65 65 70 2d 41 6c .}P..p.P ..eep-Al
0050 69 76 65 3a 20 31 31 35 0d 0a 43 6f 6e 6e 65 63 ive: 115 ..Connec
0060 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 tion: ke ep-alive
0070 0d 0a 52 65 66 65 72 65 72 3a 20 68 74 74 70 3a ..Refere r: http:
0080 2f 2f 5b 62 62 62 62 3a 3a 31 31 3a 31 33 66 66 //[bbbb: :11:13ff
0090 3a 66 65 30 30 3a 33 5d 2f 66 69 6c 65 73 2e 73 :fe00:3] /files.s
00a0 68 74 6d 6c 0d 0a 0d 0a html....
Frame (168 bytes) Reassembled TCP (454 bytes)

If a new firmware has to be flashed on the AVR Raven USB Stick, the DFU mode needs to be enabled from the Jackdaw menu by pressing the key *D*

```
*  e         Energy Scan           *
*  D         Switch to DFU mode    *
*  R         Reset (via WDT)       *
*  h,?       Print this menu       *
*                                  *
*   Fraunhofher Fokus Institute    *
**********************************
Entering DFU Mode...
cu: Got hangup signal

Disconnected.
```

When the Stick is in DFU mode, the new firmware can then be flashed:

```
sudo dfu-programmer at90usb1287 erase
sudo dfu-programmer at90usb1287 flash ravenusbstick.hex
sudo dfu-programmer at90usb1287 start
```

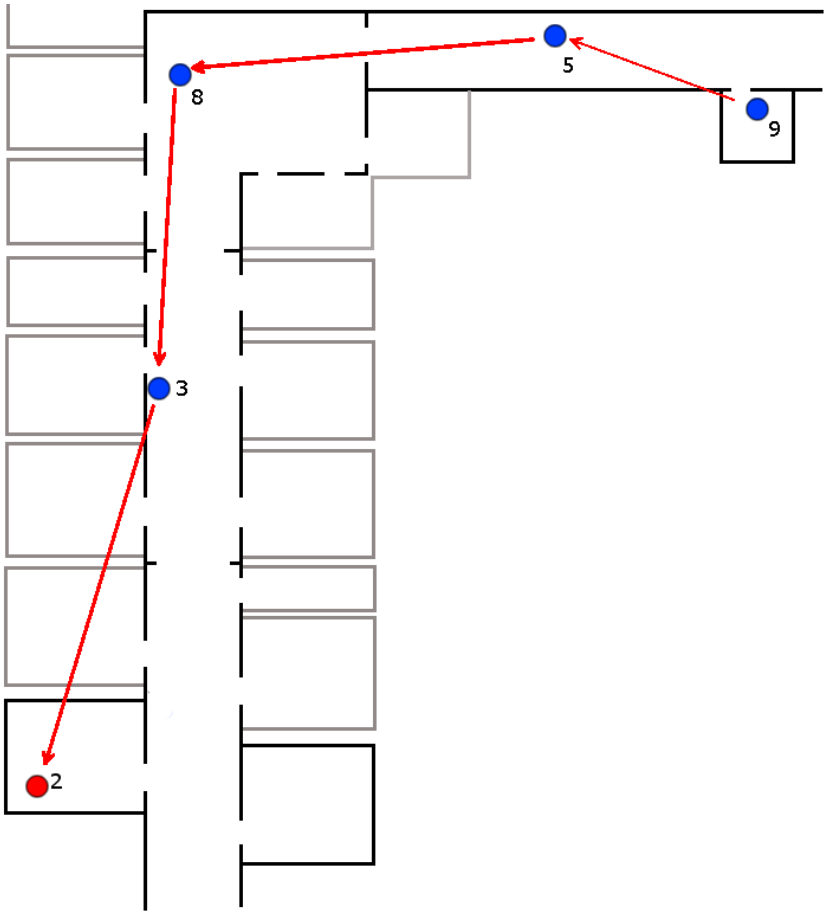
In alternative, it is possible to change the Makefile adding the following lines in *~/Contiki/examples/ravenusbstick/Makefile*

```
flash: all
      sudo dfu-programmer at90usb1287 erase
      sudo dfu-programmer at90usb1287 flash ravenusbstick.hex
      sudo dfu-programmer at90usb1287 start
```

And then simply type `$ make flash` after switching to DFU mode.

Multihop UDP - RPL Collect

The RPL-Collect example can be used in order to test the multihop packet transmission in an UDP 802.15.4 network. Node 2 (IPv6 address bbbb::2) runs the *udp-sink.c* code, while nodes 3, 5, 8 and 9 (IPv6 address bbbb::11:13ff:fe00:X)run the *udp-sender.c* code.



Sink Node fe80::11:13ff:fe00:**2**
********BOOTING CONTIKI*********
OSCAL: a:128 b:128
I am sink!
System online.
UDP server started
created a new RPL dag
Server IPv6 addresses: ::
bbbb::2
fe80::11:13ff:fe00:2
Created a server connection with remote address :: local/remote port 5688/8775
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 256
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 256
Received an RPL control message
RPL: Received a DIS from fe80::11:13ff:fe00:3
RPL: Multicast DIS => reset DIO timer
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 256
Received an RPL control message

RPL: Received a DIO from fe80::11:13ff:fe00:3
RPL: Neighbor added to neighbor cache fe80::11:13ff:fe00:3, 02:11:13:ff:fe:00:00:03
RPL: Incoming DIO rank 1536
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 640
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 256
Received an RPL control message
RPL: Received a DAO from fe80::11:13ff:fe00:3
RPL: DAO lifetime: 255, prefix length: 128 prefix: bbbb::11:13ff:fe00:3
RPL: Added a route to bbbb::11:13ff:fe00:3/128 via fe80::11:13ff:fe00:3
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:3
RPL: Neighbor already in neighbor cache
RPL: Incoming DIO rank 512
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 128
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:8
RPL: Neighbor added to neighbor cache fe80::11:13ff:fe00:8, 02:11:13:ff:fe:00:00:08
RPL: Incoming DIO rank 1792
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 768
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:3
RPL: Neighbor already in neighbor cache
RPL: Incoming DIO rank 512
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 128
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 256
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:8
RPL: Neighbor already in neighbor cache
RPL: Incoming DIO rank 768
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 256
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
RPL: Sending prefix info in DIO for bbbb::

RPL: Sending a multicast-DIO with rank 256
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:3
RPL: Neighbor already in neighbor cache
RPL: Incoming DIO rank 512
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 128
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 256
SENSOR DATA: 1314370348697 30 0 134 0 3 1 1 0 22 14617 0 0 0 0 0 512 8 512 2 131 0 0 0 0 0 0 0 0 0 0
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:3
RPL: Neighbor already in neighbor cache
RPL: Incoming DIO rank 512
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 128
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
RPL: Neighbor fe80::11:13ff:fe00:3 is known. ETX = 1
SENSOR DATA: 1314370362957 30 0 149 0 8 1 2 0 22 14617 0 0 0 0 0 768 8 768 2 131 0 0 0 0 0 0 0 0 0 0
SENSOR DATA: 1314370369985 30 0 156 0 3 2 1 0 22 17306 0 0 0 0 0 512 8 512 2 131 0 0 0 0 0 0 0 0 0 0
SENSOR DATA: 1314370377017 30 0 163 0 5 1 3 0 22 14617 0 0 0 0 0 2048 8 1024 1 131 0 0 0 0 0 0 0 0 0 0
SENSOR DATA: 1314370384045 30 0 170 0 8 2 2 0 22 17306 0 0 0 0 0 768 8 768 2 131 0 0 0 0 0 0 0 0 0 0
SENSOR DATA: 1314370398077 30 0 184 0 5 2 3 0 22 17306 0 0 0 0 0 2048 8 1024 1 131 0 0 0 0 0 0 0 0 0 0
SENSOR DATA: 1314370446113 30 0 233 0 3 3 1 0 22 26825 0 0 0 0 0 512 8 512 2 262 0 0 0 0 0 0 0 0 0 0

Multihop Messages: node 5 recives the packets from node 9 and forwards them to node 8. This is visible from the RPL debug messages.

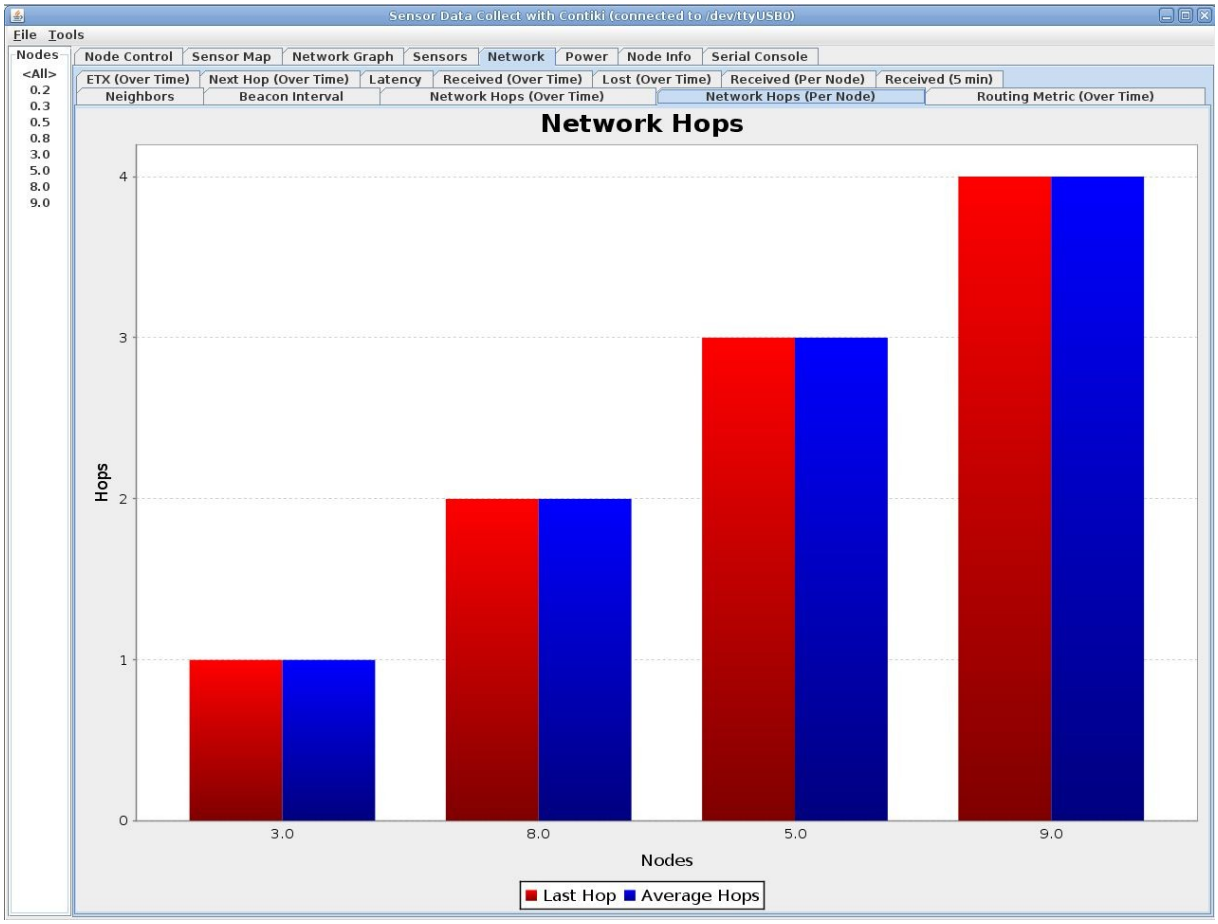
Intermediate node fe80::11:13ff:fe00:**5**

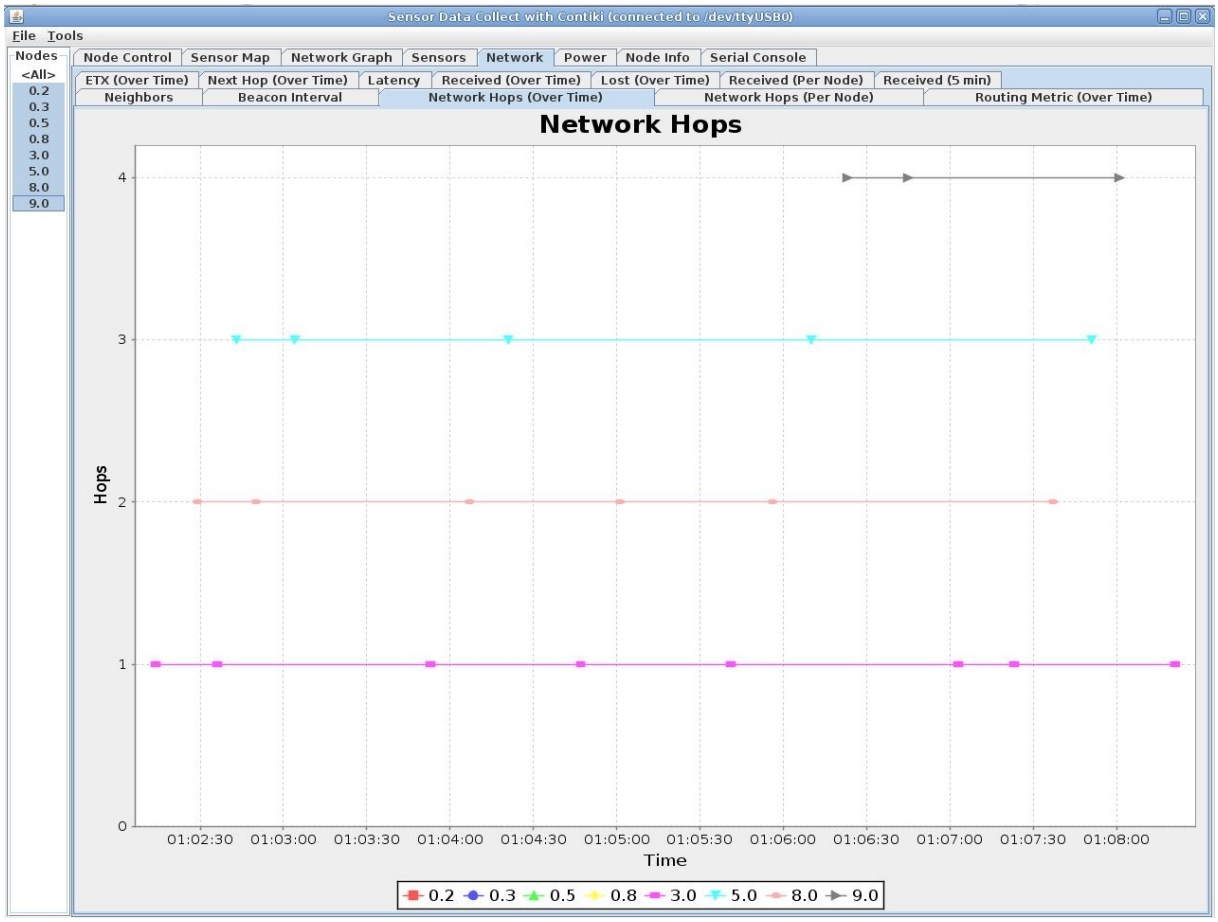
RPL: Sending prefix info in DIO for bbbb::
RPL: Sending a multicast-DIO with rank 768
Received an RPL control message
RPL: Received a DAO from fe80::11:13ff:fe00:9
RPL: DAO lifetime: 255, prefix length: 128 prefix: bbbb::11:13ff:fe00:9
RPL: Forwarding DAO to parent fe80::11:13ff:fe00:8
Received an RPL control message
RPL: Received a DIO from fe80::11:13ff:fe00:9
RPL: Neighbor already in neighbor cache
RPL: Incoming DIO rank 1024
RPL: DIO suboption 2, length: 6
RPL: DAG MC: type 7, flags 8, aggr 0, prec 0, length 2, ETX 384
RPL: DIO suboption 4, length: 14
RPL: DIO Conf:dbl=8, min=12 red=10 maxinc=768 mininc=256 ocp=1 d_l=255 l_u=65535
RPL: DIO suboption 8, length: 30
RPL: Copying prefix information
Received an RPL control message

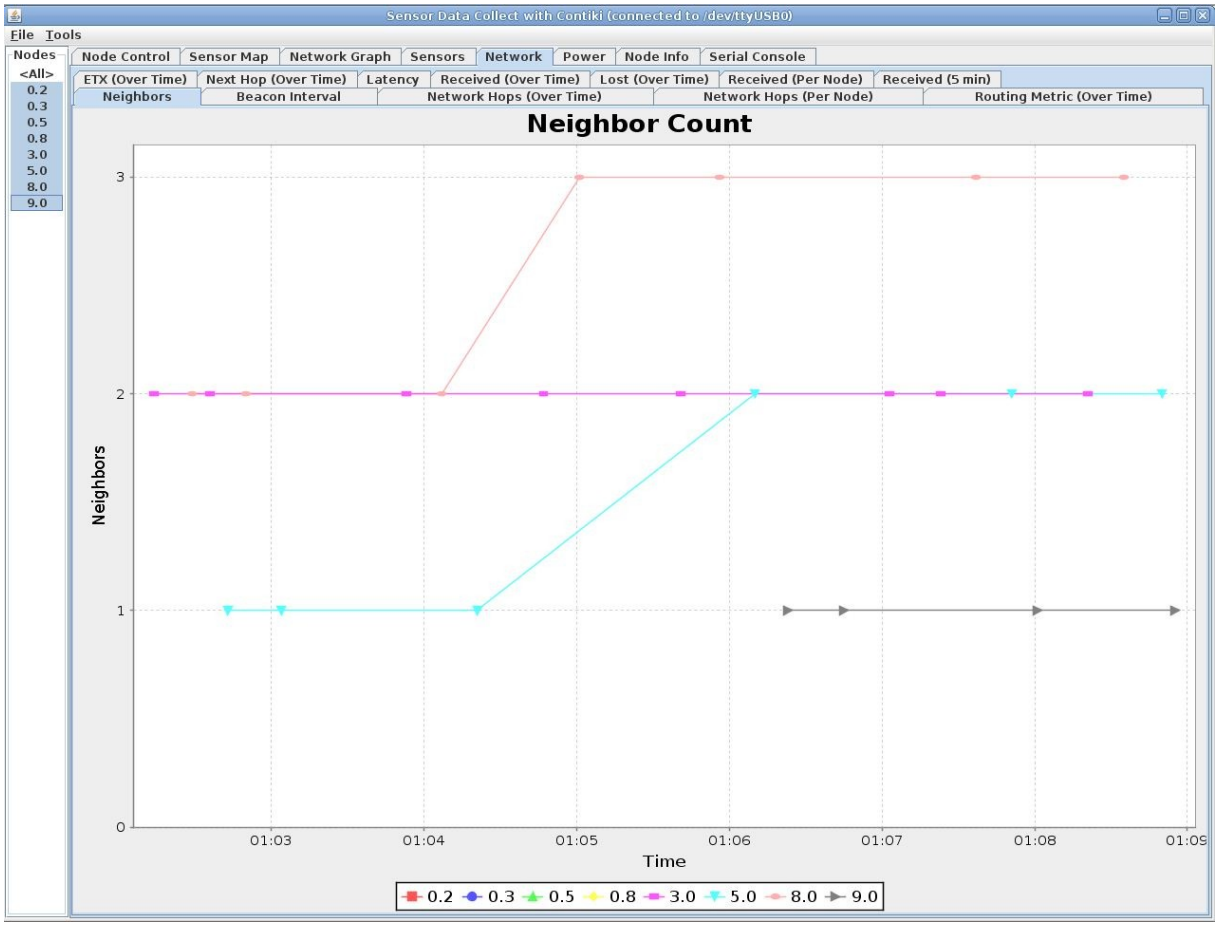
Collect View

In order to view some information about the packets transmitted, the number of hops, the beacon interval and many others, a Java application named Collect-View is available:

```
$ cd tools/collect-view/
$ ant
$ cd dist/
$ java -jar collect-view.jar /dev/ttyUSB0
```




Sensor Data Collect with Contiki (connected to /dev/ttyUSB0)
File Tools
Nodes
<All>
0.2
0.3
0.5
0.8
3.0
5.0
8.0
9.0
Node Control
Sensor Map
Network Graph
Sensors
Network
Power
Node Info
Serial Console
ETX (Over Time)
Next Hop (Over Time)
Latency
Received (Over Time)
Lost (Over Time)
Received (Per Node)
Received (5 min)
Neighbors
Beacon Interval
Network Hops (Over Time)
Network Hops (Per Node)
Routing Metric (Over Time)
Network Hops
Hops
0
1
2
3
4
01:02:30
01:03:00
01:03:30
01:04:00
01:04:30
01:05:00
01:05:30
01:06:00
01:06:30
01:07:00
01:07:30
01:08:00
Time
0.2 0.3 0.5 0.8 3.0 5.0 8.0 9.0


Sensor Data Collect with Contiki (connected to /dev/ttyUSB0)
File Tools
Nodes
<All>
0.2
0.3
0.5
0.8
3.0
5.0
8.0
9.0
Node Control
Sensor Map
Network Graph
Sensors
Network
Power
Node Info
Serial Console
ETX (Over Time)
Next Hop (Over Time)
Latency
Received (Over Time)
Lost (Over Time)
Received (Per Node)
Received (5 min)
Neighbors
Beacon Interval
Network Hops (Over Time)
Network Hops (Per Node)
Routing Metric (Over Time)
Neighbor Count
Neighbors
0
1
2
3
01:03
01:04
01:05
01:06
01:07
01:08
01:09
Time
0.2 0.3 0.5 0.8 3.0 5.0 8.0 9.0

As it is possible to see in the picture above, node 5 increases its neighbors from one (node 8) to two (node 9), while node 8 increases the number of neighbors from two (node 3 and node 5) to three (node 3, 5 and 9).