BLIP 2.0 Tutorial

From TinyOS Wiki
(Redirected from Blip 2.0 Tutorial)

Contents

BLIP 2.0 Tutorial

Before you can complete this tutorial, you should have a working TinyOS environment installed on a Linux machine. Be sure you have upgraded your toolchain; it is particularly important that you use msp430-gcc 4.5.3 since that is the tested version. It also contains a significantly improved optimizer which is required to get the edge router code to fit.

The first step to running a blip subnetwork is install PppRouter an connect to it using the Linux ppp daemon. Before doing so, it may be useful to review some of the background material in the BLIP Tutorial on IPv6 terminology.

Compile and install PppRouterC

PppRouterC is an application which has two interfaces -- one is a serial link between the mote and a computer, and the other is its 6lowpan interface communicating over 802.15.4. It acts as an edge router for BLIP 2, routing packets between the 6lowpan subnet and other networks like your PC. To compile it,

```
$ cd $TOSROOT/apps/PppRouter
$ make telosb blip
```

Then install it on a mote with ID 1 -- static addressing is default. You should then be able to initiate a ppp connection with that mote using:

```
$ pppd debug passive noauth nodetach 115200 /dev/ttyUSB0 nocrtscts nocdtrcts lcp-echo-interval 0 noccp noip ipv6
$ ifconfig ppp0 add fec0::100/64
```

Once the link is up, you need to assign your end an address on the prefix; that's what the second command does. You can control how addresses are assigned; see the Address Assignment section for more information.

UDPEcho

The basic image for motes not needing ppp is UDPEcho -- make and install it the same way

```
$ cd $TOSROOT/apps/UDPEcho
$ make telosb blip install.2 bsl,/dev/ttyUSB1
```

Since we're not using DHCP here, we give it a new address based on TOS_NODE_ID. Once you are running a PppRouter, UDPEcho, and have connected with `pppd`, you can use the udp shell to inspect the routing table:

```
$ nc6 -u fec0::2 2000
route
destination             gateway         iface
::/0            fe80::22:ff:fe00:1              pan
```

As you can see, RPL has provisioned a default route through the PPP edge router.

For more fun, you can ping the link-local all-nodes multicast group to see who your neighbors are:

```
$ nc6 -u fec0::f6 2000
ping6 ff02::1
fe80::22:ff:fe00:e0 icmp_seq=0 ttl=1 time=52 ms
fe80::22:ff:fe00:a4 icmp_seq=0 ttl=1 time=83 ms
...
fe80::22:ff:fe00:e0 icmp_seq=9 ttl=1 time=46 ms
fe80::22:ff:fe00:a4 icmp_seq=9 ttl=1 time=76 ms
10 packets transmitted, 20 received
```

Finally, if you're using DHCP the 'leases' command will show you your lease:

```
leases
lease on fec0::f6
iaid: 1 valid: 3265 t1: 3600 t2: 5400
duid: 00:01:00:01:14:7a:bc:51:00:50:8d:ca:5a:06
```

Address Assignment

BLIP either assigns IPv6 addresses statically at compile time, or dynamically using DHCPv6. It is configured to use static addressing out of the box. By default, each mote configures itself with three addresses: two link-local addresses and one global address on the prefix fec0::/64.

Static Address Mode

Static addressing is selected by defining the IN6_PREFIX macro at compile time; this is set by default for PppRouter and UDPEcho. The 802.15.4 interface is assigned three addresses:

1. Link-local address 1 is configured from the unique EUI-64 obtained from the `LocalIeeeEui64C` component. If you want to figure out what address a mote has, you can install the apps/tests/TestEui app (make epic) and then look at the printf output.
2. Link-local address 2 is chosen based on the panid and nodeid as specified in RFC4944. The default panid is 0x22, so if you install an image with ID=1, the corresponding link-local address would be fe80::22:ff:fe00:1. Note that the Unique/Local bit in the IPv6 Interface Identifier is not set for addresses derived from the 16-bit id, but is set when using the EUI-64. For instance, if my EUI-64 is 00:17:3B:00:11:0F:FD, the corresponding IPv6 link-local address is fe80::2:173B:11:0FFD. The U/L bit has been toggled.
3. Finally, a global address is configured using the prefix defined at compile time (`IN6_PREFIX`). The default prefix is fec0::/64, so if the node id is 1, the ipv6 address is fec0::1.

You can disable all global addresses by defining IN6_NO_GLOBAL in your Makefile.

Summary:

- `IN6_NO_GLOBAL` : disable global addresses, only link-local
- `IN6_PREFIX` : configure a global address using the specified prefix.

DHCPv6 Address Mode

If IN6_PREFIX is not set, blip instead assumes that you wish to assign addresses using DHCP. This requres running a DHCPv6 server somewhere in your network; I use dibbler with a small patch to reduce the size of the messages.

In this architecture, motes configure themselves only with link-local address 1 (based on the EUI-64) on boot. They send DHCPv6 solicitations to the link-local dhcpv6 all-agents multicast group to find neighboring servers or relay agents. Each router also runs a DHCPv6 relay agent, which forwards address solications up the routing tree to the edge where the DHCPv6 server is presumed to be located.

Once the DHCPv6 server has assigned the mote a global address (based on its configuration), the mote will attempt to assign itsself a 16-bit address based on the assigned address if the image is configured with `BLIP_DERIVE_SHORTADDRS`; it then starts running the routing protocol (RPL).

The only way to find out what address a mote has received is by looking in the DHCP logs; the allocation request contains the mote's EUI-64, so it is possible to configure the server to maintain a consistent mapping between EUI-64's and short addresses.

More Reading

In addition to information on BLIP 2.0 Internals, the section on the [[BLIP_Tutorial#Addressing Structures | Addressing], UDP Shell, and Network Programming are still relevant and correct except for minor changes:

- all includes `ip.h`, `lib6lowpan.h` are now in the `lib6lowpan`directory, so you must `#include <lib6lowpan/ip.h>`.
- `struct ip_metadata` has been renamed `struct ip6_metadata`.