6. Set Up The Bridge

Make sure all your network cards are working nicely and are accessible. If so, **ifconfig** will show you the hardware layout of the network-interface. If you have problems making your cards work please read the Ethernet-HOWTO at http://www.linuxdoc.org/HOWTO/Ethernet-HOWTO.html. Don't mess around with IP-addresses or net-masks. You will not need it, until you bridge fully operational an up.

After you did the steps mentioned above a **modprobe -v bridge** should show no errors. You can check the success by issuing a **cat /proc/modules**. Also for each of the network cards you want to use in the bridge the **ifconfig** **`whateverNameYourInterfaceHas`** should give you some information about the interface.

If your bridge-utilities have been correctly built and your kernel and bridge-module are OK, then issuing a **brctl** should show a small command synopsis.

6.1. brctl Command Synopsis

```
root@mbb-1:~ # brctl
commands:
    addbr           <bridge>                add bridge                      ❶
    addif           <bridge> <device>       add interface to bridge         ❷
    delbr           <bridge>                delete bridge                   ❸
    delif           <bridge> <device>       delete interface from bridge    ❹
    show                                    show a list of bridges          ❺
    showbr          <bridge>                show bridge info                ❻
    showmacs        <bridge>                show a list of mac addrs        ❼

    setageing       <bridge> <time>         set ageing time                 ❽
    setbridgeprio   <bridge> <prio>         set bridge priority             ❾
    setfd           <bridge> <time>         set bridge forward delay        ❿
    setgcint        <bridge> <time>         set garbage collection interval (11)
    sethello        <bridge> <time>         set hello time                  (12)
    setmaxage       <bridge> <time>         set max message age             (13)
    setpathcost     <bridge> <port> <cost>  set path cost                   (14)
    setportprio     <bridge> <port> <prio>  set port priority               (15)
    stp             <bridge> <state>        {dis,en}able stp                (16)
```

❶❸

The **brctl addbr** **`bridgename`** command creates a logical bridge instance with the name **`bridgename`**. You will need at least one logical instance to do any bridging at all. You can interpret the logical bridge being a container for the interfaces taking part in the bridging. Each bridging instance is represented by a new network interface.

Example 6. Creating A Instance

```
root@mbb-1:~ # brctl addbr mybridge1
```

The corresponding "shutdown" command is **brctl delbr** `bridgename`.

Caution
brctl delbr `bridgename` will only work, if there are no more interfaces added to the instance you want to delete.

❷❹

The **brctl addif** `bridgename device` command enslaves the network device `device` to take part in the bridging of `bridgename`. As a simple explanation, each interface enslaved into a instance is bridged to the other interfaces of the instance.

The corresponding command to tale a interface out of the bridge would be **brctl delif** `bridgename device`

❺

The **brctl show** command gives you a summary about the overall bridge status, and the instances running as shown in Example 7. If you are interested in detailed information about a instance and it's interfaces you will have to check ❻.

Example 7. Output Of brctl show

```
root@mbb-1:~ # brctl show
bridge name     bridge id               stp enabled
mybridge1       0000.0800062815f6       yes
```

❻

The **brctl showbr** `bridgename` command gives you a summary about a bridge instance and it's enslaved interfaces.

Example 8. Output Of brctl showbr `bridgename`

```
root@mbb-1:~ # brctl showbr mybridge1
mybridge1
 bridge id              0000.0800062815f6
 designated root        0000.0800062815f6
 root port                 0                    path cost                  0
 max age                   4.00                 bridge max age             4.00
 hello time                1.00                 bridge hello time          1.00
 forward delay             4.00                 bridge forward delay       4.00
 ageing time             300.00                 gc interval                4.00
 hello timer               0.84                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                   1.84
 flags


eth0 (1)
 port id                8001                    state                    forwarding
 designated root        0000.0800062815f6       path cost                 100
 designated bridge      0000.0800062815f6       message age timer           0.00
 designated port        8001                    forward delay timer         0.00
 designated cost           0                    hold timer                  0.84
 flags

eth1 (2)
 port id                8002                    state                    forwarding
 designated root        0000.0800062815f6       path cost                 100
 designated bridge      0000.0800062815f6       message age timer           0.00
 designated port        8002                    forward delay timer         0.00
```

```
designated cost         0                    hold timer                  0.84
flags
```

❼ The **brctl showmacs** `bridgename` command gives you a list of MAC-addresses of the interfaces which are enslaved in `bridgename`.

Example 9. Output Of brctl showmacs `bridgename`

```
root@mbb-1:~ # brctl showmacs mybridge1
port no mac addr                is local?       ageing timer
  1     00:10:4b:b6:c6:e4       no                119.25
  1     00:50:04:43:82:85       no                  0.00
  1     00:50:da:45:45:b1       no                 76.75
  1     00:a0:24:d0:4c:d6       yes                 0.00
  1     00:a0:24:f0:22:71       no                  5.81
  1     08:00:09:b5:dc:41       no                 22.22
  1     08:00:09:fb:39:a1       no                 27.24
  1     08:00:09:fc:92:2c       no                 53.13
  4     08:00:09:fc:d2:11       yes                 0.00
  1     08:00:09:fd:23:88       no                230.42
  1     08:00:09:fe:0d:6f       no                144.55
```

❽ Sets the aging time. The aging time is the number of seconds a MAC-address will be kept in the forwarding database after having received a packet from this MAC address. The entries in the forwarding database are periodically timed out to ensure they won't stay around forever. Normally there should be no need to modify this parameter.

❾ Sets the bridge's relative priority. The bridge with the lowest priority will be elected as the root bridge. The root bridge is the "central" bridge in the spanning tree. More information about STP you find at Section 7.1.

❿ Sets the forwarding delay time. The forwarding delay time is the time spent in each of the Listening and Learning states before the Forwarding state is entered.

(11) Sets the garbage collection interval. Every (this number) seconds, the entire forwarding database is checked for timed-out entries. The timed-out entries are removed.

(12) Sets the hello time. Every (this number) seconds, a hello packet is sent out by the Root Bridge and the Designated Bridges. Hello packets are used to communicate information about the topology throughout the entire Bridged Local Area Network. More information about STP you find at Section 7.1.

(13) Sets the maximum message age. If the last seen (received) hello packet is more than this number of seconds old, the bridge in question will start the takeover procedure in attempt to become the Root Bridge itself. More information about STP you find at Section 7.1.

(14) Sets the cost of receiving (or sending, I'm not sure) a packet on this interface. Faster interfaces should have lower path costs. These values are used in the computation of the minimal spanning tree. More information about STP you

find at Section 7.1. Paths with lower costs are likelier to be used in the spanning tree than high-cost paths (As an example, think of a gigabit line with a 100Mbit or 10Mbit line as a backup line. You don't want the 10/100Mbit line to become the primary line there.)

The Linux implementation currently sets the path cost of all eth* interfaces to 100, the nominal cost for a 10Mbit connection. There is unfortunately no easy way to discern 10Mbit from 100Mbit from 1Gbit Ethernet cards, so the bridge cannot use the real interface speed.

(16)
With this parameter you can enable or disable the Spanning Tree Protocol.

(12)(14)(16)
This parameters are only of interest, if you have more than one bridge in your LAN and stp enabled. Before modifying them you should read Section 7.1.

6.2. Basic Setup

The standard configuration should consist of:

1. Create the bridge interface.
```
root@mbb-1:~ # brctl addbr mybridge
```

2. Add interfaces to the bridge.
```
root@mbb-1:~ # brctl addif mybridge eth0
root@mbb-1:~ # brctl addif mybridge eth1
```

3. Zero IP the interfaces.
```
root@mbb-1:~ # ifconfig eth0 0.0.0.0
root@mbb-1:~ # ifconfig eth1 0.0.0.0
```

4. Put up the bridge.
```
root@mbb-1:~ # ifconfig mybridge up
```

5. Optionally you can configure the virtual interface **`mybridge`** to take part in your network. It behaves like one interface (like a normal network card). Exactly that way you configure it, replacing the previous command with something like:
```
root@mbb-1:~ # ifconfig mybridge 192.168.100.5 netmask 255.255.255.0 up
```

A more sophisticated setup script you will find at Example 16.

Important: If you get the terrible experience of a frozen system or some nasty behavior of your nicely shaped linux box at
```
root@mbb-1:~ # ifconfig ethn 0 0.0.0.0
```

please try (after the reboot of the system if necessary) before starting any bridge stuff at all a

```
root@mbb-1:~ # ifconfig ethn promisc up
```

If again your system is frozen it's you NIC's driver you have to blame, not the bridging code.

Prev Home Next
Preparing The Bridge Advanced Bridge Features