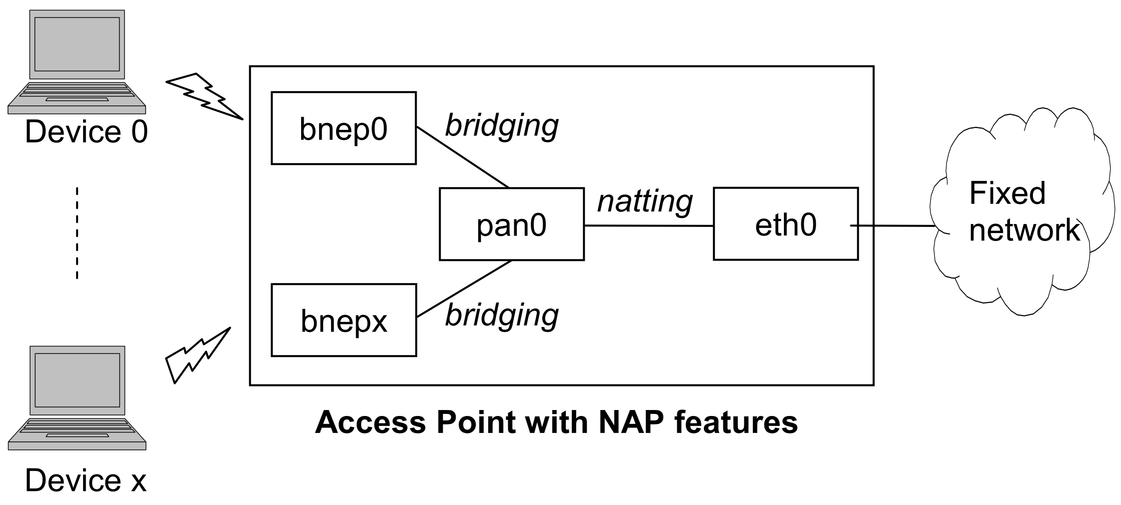
Bluetooth NAP How To

This how to describe how to set up a deamon (the Network Access Point Daemon, napd), developed by the mobilab group, useful in nomadic computing environment, where a set of mobile devices is connected to the fixed infrastructure via Access Points (Bluetooth).

The napd must be executed on the AP in order to provide the access to the fixed network infrastructure by mobile bluetooth users. The daemon set up PAN channels necessary for the use of IP over L2CAP bluetooth channels: it wait for PAN connection requests and, on accepted requests, create an IP network interface for the communication. It perform also the natting (Network Address Translation), allowing mobile device to use private IP network addresses and enabling the use of DHCP server in every cell (the zone covered by an AP).

More specifically, the IP interfaces are created using BNEP (Bluetooth Network Encapsulation Protocol): when a PAN connection is made, the Bluetooth PAN daemon (distributed with the official linux Bluetooth protocol stack: BlueZ) create an ad-hoc IP interface named bnepx (for the x-th connection). Thus, the pan daemon create an interface for each connection. In order to allow the communication through an unique virtual interface, enabling in that way the NAT service, it must be created a new virtual interface (named pan0) and merge it with the bnepx interfaces created by PAN modules. Those functionalities are provided by the napd using the bridge control modules, provided by linux distributions. In other words, differently to the PAN daemon, our NAP daemon expose an unique interface to the devices, which can always use the IP address of pan0 as default gateway, rather than use a different IP address for each connection. The various bnepx interfaces created for each connection are bridged with the virtual pan0 interface through the bridge control utilities. The IP datagrams are then forwarded to the fixed network via a normal eth0 interface (using natting). Figure below can clarify this concepts. More information about the setup of bridging functionalities within the Bluetooth PAN profile can be found at: `bluez.sourceforge.net/contrib/HOWTO-PAN`.

Below, we list the requirements, configuration notes and usage of the napd, referring to our tests on the daemon.



Access Point with NAP features

1. AP System Requirements:

- Mandrake Linux 9.1 distribution, Kernel version great or equal to 2.4.19;
- Bluetooth USB Dongle;
- Ethernet interface.

2. AP Software Requirements:

- **BlueZ packages** with PAN and sdp support: see the installation notes at www.mobilab.unina.it/bluetooth.

- **Bridge control**: In order to install the bridge control functionalities, it's needed to reconfigure the kernel: in the directory containing the kernel source and the kernel's Makefile, digit:

```
  # make menuconfig
```

  then select “networking options” -> “802.1d Ethernet Bridging” as kernel module (<M>). The modules can be installed with the commands:

```
# make modules
# make modules_install
```

  Finally, it's needed to install a control package, named as "bridge utils", useful for the control of bridging functionalities. The package can be downloaded from bridge.sf.net . After downloaded the package, extract the files with:

```
# tar zxvf <package_name>
```

  and install doing the following:

```
# ./configure
# make
# make install
```

  IMPORTANT NOTE: After the installation, please control where the program brctl has been installed with the command:

```
# whereis brctl
```

  if the program isn't localized in `/usr/local/bin`, you have to change appropriately the macro BRCTL in the scripts napd, makebridge and bridge-attach provided with our code in the `./scripts` directory.

- **DHCP Daemon**: the NAP can assign automatically the IP address to the devices by installing a DHCP daemon, downloadable from ftp.isc.org/isc/dhcp. The installation can be done as we described previously for the bridge utils. After the installation, you must create a configuration file (named `/etc/dhcpd.conf`). Below is shown an example of such a file:

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers 192.168.0.254;
option domain-name-servers 192.168.0.1, 192.168.0.2;

subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.10 192.168.0.100;
   range 192.168.0.150 192.168.0.200;
}
```

You can set the parameters (i.e lease time or addresses ranges) as you need for your purposes. Furthermore, you must create an empty file, named `/var/state/dhcp/dhcpd.leases`. An empty file can be created doing the following:

```
# touch <nome file>
```

Finally, the dhcp daemon can be started with verbose mode and listening on pan0 interface with the command:

```
# dhcpd –d –f pan0
```

- **IP forwarding and masquerading support**: the forwarding and masquerating (IPMASQ) allow IP packets routing and address translating, providing AP with NAT functionalities. For kernels 2.4.x, you have to check if the IPMASQ is present as module or statically compiled, doing the following:

  - Run the command `ls /proc/sys/net/ipv4`. These items are required and should be present regardless if your kernel built IPMASQ as modules or statically:

    - `ip_dynaddr, ip_forward`

  - To check if IPMASQ was compiled statically into the kernel, run the command `/sbin/lsmod` and see if and modules like the ones shown below at the next point. No? Ok, now run the command `ls /proc/net/` and see if you see additional /proc files such as:

    - `ip_masquerade, ip_conntrack, ip_tables_names`

    If you see these /proc entries and there WEREN'T any kernel modules loaded (shown via the "lsmod" command mentioned above), then your kernel has the IPTABLES subsystem statically compiled into it and is ready to go to use IPMASQ on this system.

- If your kernel uses IPTABLES via modules, most of the stuff listed above should have been missing (because the modules probably aren't loaded). Run the command `ls /lib/modules/`uname -r`/kernel/net/ipv4/netfilter/` where you should see files like:

  - `ip_conntrack.o, ip_conntrack_ftp.o, ip_conntrack_irc.o, ip_nat_ftp.o, ip_nat_irc.o`
    `ip_tables.o, ipt_MASQUERADE.o, iptable_nat.o, iptable_mangle.o, iptable_filter.o`

  If you see those kernel files, IPTABLES was compiled using modules and things look ready to go to use IPMASQ on this system. Further information can be found at www.ecst.csuchico.edu/~dranch/LINUX/ipmasq/c-html.

Finally, you have to check if the program "iptables" is installed on your system (try to launch it in root mode). If it isn't installed, you can download it from www.netfilter.org, then install it with the same procedure shown about the bridge utils.
For IPMASQ configuration, see next section.

3. IP masquerating configuration

For the modules loading and setup of IPMASQ, you can create and launch a script file like the following one:

```
#!/bin/sh
#
echo -e "\n\nLoading simple ipmasq rules"

# Location of the iptables and kernel module programs
#
#   If your Linux distribution came with a copy of iptables,
#   most likely all the programs will be located in /sbin.  If
#   you manually compiled iptables, the default location will
#   be in /usr/local/sbin
#
# ** Please use the "whereis iptables" command to figure out
# ** where your copy is and change the path below to reflect
# ** your setup
#
#IPTABLES=/sbin/iptables
IPTABLES=/usr/local/sbin/iptables
DEPMOD=/sbin/depmod
MODPROBE=/sbin/modprobe

# Setting the EXTERNAL and INTERNAL interfaces for the network: in our
# case, eth0 is the external interface and pan0 is the internal virtual
# interface created via brctl
#
EXTIF="eth0"
INTIF="pan0"
echo "   External Interface:  $EXTIF"
echo "   Internal Interface:  $INTIF"

echo -en "   loading modules: "
```

```
# Need to verify that all modules have all required dependencies
echo "  - Verifying that all kernel modules are ok"
$DEPMOD -a

#Load the main body of the IPTABLES module - "iptable"
#
echo -en "ip_tables, "
$MODPROBE ip_tables

#Load the stateful connection tracking framework - "ip_conntrack"
#
echo -en "ip_conntrack, "
$MODPROBE ip_conntrack

#Load the FTP tracking mechanism for full FTP tracking
#
echo -en "ip_conntrack_ftp, "
$MODPROBE ip_conntrack_ftp

#Load the IRC tracking mechanism for full IRC tracking
#
echo -en "ip_conntrack_irc, "
$MODPROBE ip_conntrack_irc

#Load the general IPTABLES NAT code - "iptable_nat"
#  - Loaded automatically when MASQ functionality is turned on
#
#  - Loaded manually to clean up kernel auto-loading timing issues
#
echo -en "iptable_nat, "
$MODPROBE iptable_nat


#Loads the FTP NAT functionality into the core IPTABLES code
#
echo -en "ip_nat_ftp, "
$MODPROBE ip_nat_ftp

echo -e "   Done loading modules.\n"

#CRITICAL:  Enable IP forwarding since it is disabled by default since
#
echo "   Enabling forwarding.."
echo "1" > /proc/sys/net/ipv4/ip_forward


# Enable dynamic IP users:
#
echo "   Enabling DynamicAddr.."
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

# Enable simple IP forwarding and Masquerading
#
echo "   Clearing any existing rules and setting default policy.."
$IPTABLES -P INPUT ACCEPT
$IPTABLES -F INPUT
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -F OUTPUT
$IPTABLES -P FORWARD DROP
$IPTABLES -F FORWARD
$IPTABLES -t nat -F
```

```
echo "FWD: Allow all connections OUT and only existing ones IN"
$IPTABLES -A FORWARD -i $EXTIF -o $INTIF -m state --state \
         ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT
$IPTABLES -A FORWARD -j LOG

echo "   Enabling SNAT (MASQUERADE) functionality on $EXTIF"
$IPTABLES -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE

echo -e "Configuration done.\n"
```

The script can be loaded automatically at system's start-up saving it as `/etc/rc.d/rc.firewall`.

4. napd compilation and execution

After extracted our source files and scripts, change the current directory to the one containing the Makefile and the subdirectories src and scripts. Thus the nap daemon can be compiled with the command

```
# make
```

The make will create an executable file in the current directory named napd (and also the .o files, removable using `make clean`). You can execute it in different manners (with root privileges) from the current directory:

```
# ./napd          (launch the napd as a daemon)
```
or:
```
# ./napd -n       (launch the napd as a normal process)
```
or:
```
# ./napd -v       (launch the napd as a normal process, verbose mode)
```

NOTE: Before executing the napd, it is needed to activate the Bluetooth interface and the service discovery protocol daemon, with the commands (in root mode):

```
# hciconfig hci0 up
# sdpd
```

You have also to execute the `makebridge` script, provided with our source file and scripts in the `scripts` folder:

```
# ./scripts/makebridge
```

such a script uses the bridge control functionalities to create the pan0 interface as a bridge.
Furthermore, you have to copy by hand the script file `bridge-attach` (provided with our source files and scripts in the `scripts` folder) in the directory `/etc/bluetooth`. You should even check if that script have the privileges of execution in root mode... briefly, do the following:

```
# cp ./scripts/bridge-attach /etc/bluetooth/.
# chmod 744 /etc/bluetooth/bridge-attach
```

On incoming requests, the daemon will create bnepx interfaces and will merge them with pan0 through the `bridge-attach` script.

5. napd installation

The napd can also be installed, in order to be executed wherever you are working in the file system, and providing an user level interface simpler to use than using directly the executable file like we have shown in previous section. Once the compilation took place (with the make command), you can install the daemon with:

```
# make install
```

The installation will move the executable napd file in /opt directory (you can change this directory by editing the makefile and change the EXECUTABLE macro), and will copy the napd script in the /sbin folder. The napd script is a wrapper offering a simple interface for use the daemon. You can start the daemon wherever directory you are (and in root mode) by editing:

```
# napd start
```

and you stop it in the same fashion, with `napd stop`.
To see what the daemon is doing, you can monitor the system log file, for example by the command:

```
# tail /var/log/messages
```

NOTE: after the napd is installed and before executing it, you have only to activate the bluetooth interface with `hciconfig hci0 up`. There is no need to launch neither the sdpd, nor the makebridge script or to copy the bridge-attach script in other folders.

Eventually, you can uninstall the daemon with:

```
# make uninstall
```