How to connect/configure

Sangoma S518 ADSL card to DSL line

for FreeBSD/OpenBSD OS.

Author: Alex Feldman

Version 1.1

October 15, 2003

The FreeBSD/OpenBSD ppp program has supported PPPoE. The goal of this article is to walk you through the steps involved in setting up a complete configuration allowing you to connect your FreeBSD/OpenBSD computer to the Internet via ADSL.

Pre-requisites

Before diving head first into the technicalities of PPPoE on FreeBSD/OpenBSD, let's make sure some basics are met. To complete the setup detailed in this article, you'll need:
a working DSL line, i.e. activated by the provider selling you DSL service.
Sangoma S518 ADSL card used to connect your computer to the DSL.

- a PPP account (authenticated with PAP or CHAP), also activated by the provider.
- a computer running FreeBSD 4.0 or later. OpenBSD
- Sangoma S518 ADSL card used to connect your computer to the DSL.
- a PPP account (authenticated with PAP or CHAP), also activated by the provider.
- a computer running FreeBSD 4.0 or later.

Kernel configuration

The kernel you are going to be using for this project must be configured with:

```
pseudo-device tun
```

And

```
options NETGRAPH          (only for FreeBSD)
```

This is not the case for the GENERIC kernel included in FreeBSD 4.0; you will have to rebuild a kernel in order to be able to use PPPoE. Please refer to the FreeBSD Handbook, section Configuring the FreeBSD Kernel for help in doing this.
Once your kernel is rebuilt with these options, make sure the corresponding tun devices are present on your system:

```
$ ls -l /dev/tun?
crw-------  1 uucp  dialer   52,   0 May 17 10:59 /dev/tun0
crw-------  1 uucp  dialer   52,   1 Nov 19  1996 /dev/tun1
crw-------  1 uucp  dialer   52,   2 Nov 19  1996 /dev/tun2
crw-------  1 uucp  dialer   52,   3 Nov 19  1996 /dev/tun3
```

If not, they can be created with:

```
# cd /dev
# ./MAKEDEV tun{0,1,2,3}
```

PPP configuration

Okay, now let's do it. First make a backup copy of the existing /etc/ppp/ppp.conf file, and then edit it to conform to follow example:

```
#
# ppp.conf: PPPoE configurtion
#
default:
```

```
# PPP over Ethernet
set device "!/usr/sbin/pppoe –i wpaadsl0"   ← Only for OpenBSD
set device PPPoE:wpaadsl0:                  ← Only for FreeBSD
set speed sync
set mru 1492
set mtu 1492
set ctsrts off
# Monitor line quality
enable lqr
# Log just a bit
set log phase tun
# insert default route upon connection
add default HISADDR
# download /etc/resolv.conf
enable dns
set authname USERNAME
set authkey PASSWORD
```

USERNAME and PASSWORD above should be replaced with your actual username and password, respectively. Also replace wpaadsl0 with your actual Sangoma ADSL interface.
The interface wpaadsl0 used by PPPoE must be UP, i.e. enabled. It is only used as transport, and does not need to be assigned an address. Eg:

```
$ ifconfig wpaadsl0
wpaadsl0: flags=8843< UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether  00:77:77:77:78:1a
        media: Ethernet autoselect
        status: Connected
```

If the flags section above doesn't list UP, bring the interface up with:

```
# ifconfig wpaadsl0 up
```

and you should be all set for the next step.

Running PPP for the first time

We are going to make a first attempt at running ppp without authentication. This test will only verify your DSL configuration and will not create a working connection yet. See the follow example:

```
# /usr/sbin/ppp
Working in interactive mode
Using interface: tun0
ppp ON bsd> set log +debug
ppp ON bsd> show physical
Name:                deflink
State:               closed
```

Device: N/A
Link Type: interactive
Connect Count: 1
Queued Packets: 0
Phone Number: N/A

Defaults:
Device List: "!/usr/sbin/pppoe –I wpaadsl0" ← only for OpenBSD
Device List: "PPPoE:wpaadsl0" ← only for FreeBSD
Characteristics: sync, cs8, no parity, CTS/RTS off
CD check delay: device specific

Connect time: 0:00:00
0 octets in, 0 octets out
Overall 0 bytes/sec

ppp ON bsd> *dial*
ppp ON bsd>
Ppp ON bsd>
Ppp ON bsd>
Ppp ON bsd>

Note how the first P of the ppp prompt becomes capitalized, indicating that the LCP phase was successfully completed. If this first P fails to become capitalized, you need to look at /var/log/ppp.log and figure out what went wrong. Since debugging output was activated with set log +debug above, there should be lots of information there. Sift through the file and pinpoint the exact point of failure.
Note how the three "P" of the prompt are capitalized, indicating that you have successfully completed the LCP phase (first P), you are successfully authenticated (second P), and your machine were assigned an IP address (third P). This constitutes the complete PPP connection process, and your machine is now online. Woo-hoo!

Testing

Now that you are successfully connected, we can start running some tests on your connection. Suspend your ppp process with ^Z, and have it run in the background with bg (or just switch to another terminal). Then run ifconfig again:
$ *ifconfig tun0*
tun0: flags=8051< UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1492
inet LOCAL_ADDR→ PEER_ADDR netmask 0xffffffff
Opened by PID 512
PEER_ADDR above is the actual address of the next device in your network path, usually it is your ISP's router. LOCAL_ADDR is your actual IP address, the one assigned to your FreeBSD/OpenBSD machine by the ISP.

Pinging the remote end of the link

First you need to make sure that you can ping the PEER_ADDR. Failing this test would indicate serious trouble with your configuration, most probably DSL-related, in which case you will have to talk to your ISP to resolve them.

```
$ ping PEER_ADDR
PING PEER_ADDR (PEER_ADDR): 56 data bytes
64 bytes from PEER_ADDR: icmp_seq=0 ttl=64 time=18.062 ms
64 bytes from PEER_ADDR: icmp_seq=1 ttl=64 time=17.115 ms
^C
--- PEER_ADDR ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 17.115/17.588/18.062/0.474 ms
```