

FreeBSD + Soekris*

Krzysztof Kozłowski[†]

13 stycznia 2006 roku

Spis treści

1 Wstęp

Projekt polegał na instalacji FreeBSD na maszynie Soekris net4801 oraz próbie przystosowania jej do roli routera. Zamiast instalować pełnego FreeBSD można skorzystać z NanoBSD lub TinyBSD. W projekcie użyłem zwykłego FreeBSD 6.0.

2 Dostęp do czystego Soekrisa

Dostęp do maszyny bez zainstalowanego żadnego systemu operacyjnego uzyskamy poprzez kabel szeregowy RS. Ważne jest, żeby ustawić tę samą prędkość transmisji, jaka widnieje w BIOS-ie Soekrisa. Przykładowy wpis w pliku */etc/remode* (prędkość - 19200 bodów) :

```
sio0|com1:dv=/dev/cuad0:br#19200:pa=none:
```

3 Serwer DHCP

Przykładowa konfiguracja */usr/local/etc/dhcpd.conf* serwera DHCP (ISC DHCP server) :

```
host soekris {  
    hardware ethernet 00:00:12:34:56:78;  
    fixed-address s01v;  
    filename "boot/pxeboot";  
}
```

*Dokument udostępniany bezpłatnie. Możliwe jest jego dowolne rozprowadzanie, ale bez zmiany zawartości. Copyright (C) 2006 Krzysztof Kozłowski

[†]Autor jest studentem informatyki na wydziale Elektrycznym Politechniki Warszawskiej.

Kontakt: k.kozlowski [at] iem.pw.edu.pl, strona: <http://acn.waw.pl/koziol>

```
option root-path "/tftpboot/boot" ;
}
```

4 Serwer TFTP i pliki do bootowania

Aby zabootować instalatora FreeBSD 6.0 potrzebne będą niektóre pliki z płyty *6.0-RELEASE-i386-bootonly*. Zakładając, że drzewo serwera TFTP to */tftpboot/*, a płyta zamontowana została w */cdrom*, wykonujemy :

```
# mkdir /tftpboot/boot
# cp -pR /cdrom/boot/* /tftpboot/boot
```

oraz edytujemy plik */tftpboot/boot/loader.conf* wpisując do niego :

```
mfsroot_load="YES"
mfsroot_type="mfs_root"
mfsroot_name="/boot/mfsroot"
console="comconsole"
verbose_loading="YES"
comconsole_speed="19200"
```

5 Rekompilacja pxeboot+loader

Niestety FreeBSD 6.0 używa emulacji terminala cons25 co uniemożliwia odczyt czegokolwiek po RS-ie z Soekrisa. Należy przekompilować *pxeboot* oraz *loader* :

```
# cd /usr/src/sys/boot/i386/libi386
# vi Makefile
  USUNĄĆ LINIĘ :
  CFLAGS+= -DTERM_EMU
# cd /usr/src/sys/boot
# make clean; make depend; make;
# cp i386/pxeldr/pxeboot /tftpboot/boot
# cp i386/loader/loader /tftpboot/boot
```

6 Rekompilacja jądra

Standardowy kernel GENERIC próbuje zamontować zdalny korzeń poprzez NFS, a w nas interesuje protokół TFTP. W tym celu należy samodzielnie skompilować jądro. Diff z przykładowego pliku konfiguracyjnego kernela w stosunku do GENERIC :

```

-cpu          I486_CPU
  cpu         I586_CPU
  cpu         I686_CPU
-ident        GENERIC
+ident        SOEKRIS
+options      CPU_GEODE
+options      CPU_SOEKRIS
-options      NFSCLIENT
-options      NFSSERVER
-options      NFS_ROOT

```

Przy czym można od razu bardziej odchudzić go, aby później wykorzystać jako jądro dla zainstalowanego systemu. Opcje do usunięcia : *UFS_ACL*, *MSDOSFS*, *CD9660*, *GEOM_GPT*, *KTRACE*, *XSERVER*. Sterowniki do usunięcia : *apic*, wszystkie SCSI, RAID, Wireless, USB, Firewire.

Instalacja jądra :

```

# cd /tftpboot/boot/kernel
# mv kernel kernel.old
# cp /tmp/obj/usr/src/sys/SOEKRIS/kernel ./kernel

```

7 Instalacja

Po tych wszystkich czynnościach można przejść do instalacji FreeBSD, która w przypadku Soekris zasadniczo nie będzie się różniła w żaden sposób od typowej. Warto pamiętać o ograniczonej przestrzeni kart flash i wgrać system w całkowicie podstawowej konfiguracji.

8 Czynności poinstalacyjne

Bezpośrednio po instalacji, ale przed resetem maszyny, należy uruchomić skonfigurować system, tj. :

1. Edycja */etc/ttys* i modyfikacja *ttyd0* do postaci "*ttyd0 "/usr/libexec/getty std.19200" vt100 on secure*", a także zakomentowanie linii *ttyvX* (nie będą używane).
2. Włączenie usługi *sshd*

Po przejściu do shella (FIXIT) musimy :

1. Pobrać z serwera poprawne pliki *loader* oraz *pxeboot* :

```

# cd /boot
# tftp 10.0.0.1
# get /boot/loader
# get /boot/pxeboot

```

2. Wyedytować plik `/etc/ssh/sshd_config`, aby pozwolić rootowi na logowanie.

Jeżeli nasza maszyna zostanie w pełni skonfigurowana, to powinniśmy w pliku `/etc/fstab` ustawić tryb `ro` dla wszystkich partycji znajdujących się na pamięci flash. Po restarcie system sam utworzy ramdysk dla katalogu `/var` i `/tmp`.

9 Wydajność Soekris

Soekris net4801 w laboratorium wyposażony był w 128 MB pamięci oraz procesor klasy Pentium MMX 233 MHz. O ile ilość RAM-u uznać można za całkowicie wystarczającą do wyznaczonej mu roli routera, to okazało się, że wydajność nie była zadowalająca.

9.1 Mostek test

Przeprowadzony został test przepustowości urządzenia w przypadku pracy jako prosty mostek. Teoretycznie nie występuje wtedy zapotrzebowanie na pamięć RAM, a znaczenie dla końcowej wydajności ma przede wszystkim magistrala PCI łącząca karty sieciowe oraz procesor. W sieci 100 mbitowej udało się uzyskać transfer rzędu **7 MB/s**, ale użycie procesora Soekrisa wynosiło praktycznie **100%** (wszystko przeznaczone na *interrupt*. Stanowi to około **70%** przepustowości sieci (bez pośrednictwa Soekrisa klient osiągnął transfer ok. 10 MB/s). Komunikat : "*Interrupt storm detected on irq10: sis0 sis1+"; throttling interrupt*" na wyjściu Soekrisa potwierdza pewne problemy wydajnościowe Soekrisa.

9.2 Router/NAT test

Próba pracy Soekrisa jako NAT wypadła mniej zadowalająco. Klient za NAT-em osiągnął maksymalny ciągły transfer (protokół FTP) ok. **1 MB/s**, co stanowi tylko **10%** przepustowości sieci. Użycie czasu CPU wyglądało inaczej niż w przypadku mostka - **30%** *interrupt* i **60%** *system* (firewall miał tylko podstawowe reguły).

9.3 Optymalizacja

Aby poprawić wydajność pracy Soekrisa w wyżej wymienionych zastosowaniach warto spróbować następujących rozwiązań.

Polling Zamiast typowych przerw sprzętowych użyć *polling(4)*. Szczególnie owocne może to się okazać na maszynie z ograniczoną wydajnością.

1. Technika ta wspierana musi być przez sterownik karty sieciowej - w przypadku Soekrisa jest to *sis(4)* i spełnia on to kryterium.
2. Kernel musi zostać skompilowany z *options DEVICE_POLLING*.
3. Zmienna MIB *kern.polling.enable* musi mieć wartość 1.

Netgraph NAT Od demona *natd* pod względem wydajności będzie lepszy dedykowany moduł kernela/ipfw. Rozwiązanie takie oferuje *netgraph(3)* - *ng_nat(4)*. Użycie go zamiast *natd* dało wzrost przepustowości o 75%, do wartości **1,75 MB/s**. Aby użyć *ng_nat* (*X.X.X.X* - adres IP publicznego interfejsu) :

```
# kldload netgraph
# kldload ng_ipfw
# kldload ng_nat
# ngctl mkpeer ipfw: nat 60 out
# ngctl name ipfw:60 nat
# ngctl connect ipfw: nat: 61 in
# ngctl msg nat: setaliasaddr X.X.X.X
# sysctl net.inet.ip.fw.one_pass=0
```

a zamiast reguł *divert* w *ipfw* należy użyć :

```
ipfw add 300 netgraph 61 all from any to any in via $public_interface
ipfw add 400 netgraph 60 all from any to any out via $public_interface
```

10 Literatura

1. <http://www.ultradesic.com/index.php?section=22>
2. <http://www.soekris.com/>
3. <http://www.soekris.com/support.htm>