Building a FreeBSD Appliance With NanoBSD

Poul-Henning Kamp

phk@FreeBSD.org

Appliance:

- A device or instrument designed to perform a specific function, especially an electrical device, such as a toaster, for household use.

Computer appliance:

- A specialized server that is designed for ease of installation and maintenance. Appliances have their hardware and software bundled in the product, so all applications are pre-installed. The appliance is plugged into an existing network and can begin working (almost) immediately.

Appliance example (1)

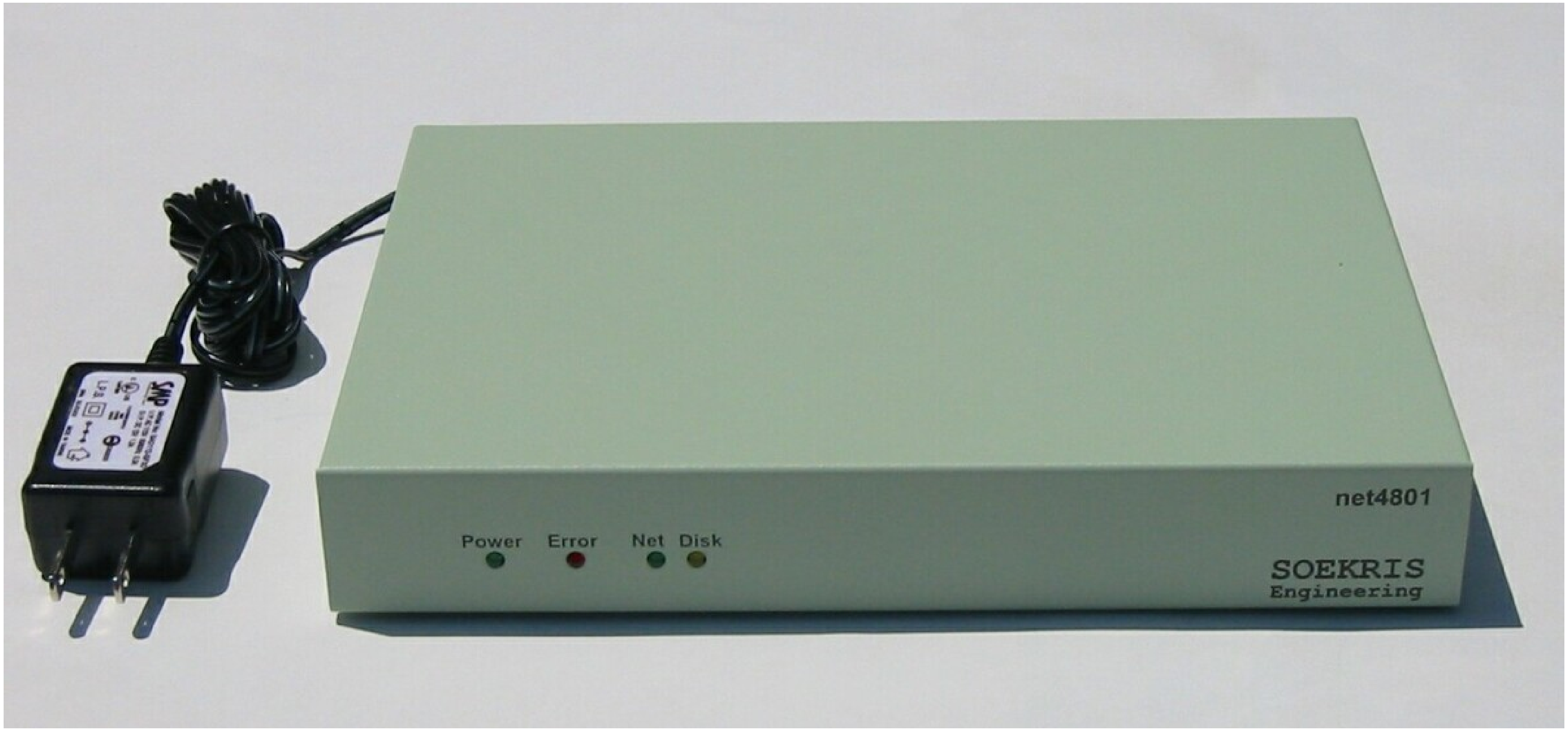
- Firewall / Router

Appliance example (2)

- VPN gateway

Appliance example (3)

- Data Collection Engine



Appliance Design

- Hardware considerations
  - Interfaces
  - Performance
  - Moving parts vs. Solid state
  - Power demand (& cost of ownership)
  - Physical (temperature, size, vibration)
  - Cost

Rotation: just say no!

- #1 cause of failure in embedded systems:

  “Something stopped rotating”

- Disks will crash.
  - And generate heat.
- Fans will fail.
  - And fill the interior with dust.

Rotation: just say no!

- #1 cause of failure in embedded systems

“Something stopped rotating”

- Disks will crash.
  - And generate heat.
- Fans will fail.
  - And fill the interior with dust.

Disks

- Disks die from old age and heat
  - Some die young
- Laptop disks are tougher than desktop disks
- Beware of “duty cycle”
  - Many disks only designed for 8h/d duty
- Disks need civilized temperatures
  - 10 - 40 °C
- Disks don't like vibration and shock

Flash storage

- Indestructible in read-only usage
- Flash cells die after N erase operations
- Flash adaptation layer
  - Wear leveling distributes the damage
  - Bad sector handling tries to recover
  - Bigger media last longer than small media
  - Good, but not perfect remedy.

Counting flash writes

- 200.000 writes, worst case:
  - Superblock updated 1/s = 55 hours lifetime
  - File written 1/m = 138 days lifetime
  - File written 1/h = 22 years lifetime
  - File written 1/d = 547 years lifetime

Flash Adaptation Layer

- Attempt to wear out flash cells equally fast
- Logical->Physical mapping
- Bad bit/sector handling
- Works much better if you tell it about erase
- ...or do large sequential writes (camera)

FAL's MTBF

- Conservative rule of thumb:
  - Multiply write guarantee by free/used ratio.

- Example:
  - 200k writes per cell
  - 100MB used on 4GB media
    - 200k * (4GB-100MB)/100MB =
      200k * 39 =
      7.8 Mwrites per logical sector.

Industrial Flash ?

- Expensive
- Guaranteed number of writes
  - Typical: 10x commercial (~2M writes per cell)
- Good EMC protection
- -40...+70 °C

Consumer Flash

- Cheap
- Vague promises of durability
  - 20k…200k writes
    - Before or after FAL ?
    - Guaranteed or worst case ?
    - Per sector or per photo ?
- Often faster than industrial flash
- Easy to get hold of

Power budget

- 1W * 24h * 365d * .25 $/kWh ~= 2$/Wy

Machine	**Watt [avg]**	**USD/year**
Real Server	220	440
Light serve	120	240
Desktop	60	120
Mini-ITX	30	60
Soekris	4	8

- Power-over-ethernet: max 12.95W

4W = Power options

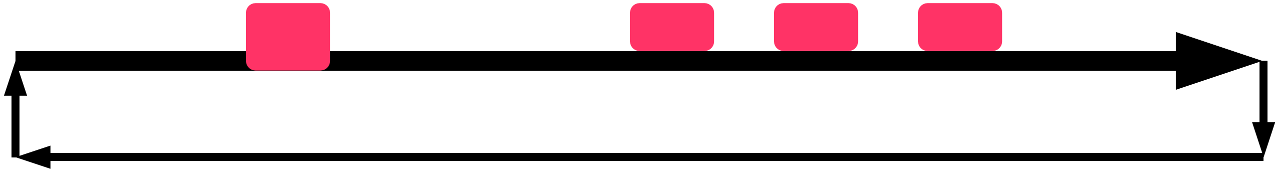
- Solar power
  - Depends on your lattitude/climate.
- Battery backup
  - $100 Lead-Acid -> 2+ days without power.
- Portable
  - Runs on 6 D-size batteries
- Car/motorcycle
  - Remember surge-protection.

Human Factors

- User interaction when no screen available
  - telnet/ssh into box
  - HTML/webserver interface
  - Serial console
- Hardware solutions
  - LED
  - LCD displays

Flashing a hint

- The LED(4) device driver can be used to signal using a LED, lamp, foghorn etc.
  - `/bin/echo “d13” > /dev/led/error`



More LED(4) features

-
  ```
  /bin/echo 0 > /dev/led/error
  ```

-
  ```
  /bin/echo 1 > /dev/led/error
  ```

-
  ```
  /bin/echo f > /dev/led/error
  ```

-
  ```
  /bin/echo sAaAaAcEaEaEcAaAaA \
     > /dev/led/error
  ```

-
  ```
  /usr/games/morse -l \
     “+++ OUT OF CHEESE +++” \
     > /dev/led/error
  ```

LCD displays

- Readily available with usable interfaces
  - Serial, USB, parallel, etc
- Expensive compared to computer
  - Remember to check eBay for bargains
- Mechanical/Mounting issues
- Also need a keyboard/keypad

Cables and interfaces

- Beware of ground-loops
  - serial/parallel/usb/gpio/power
  - (Non-POE) Ethernet is isolated.
- GPIO pins are sensitive
  - Surgeprotection on inputs
  - Drivers on outputs
- Lightning protection if outdoors.
  - In particular POE!

What is NanoBSD ?

- NanoBSD is just FreeBSD

- Compiled from FreeBSD source tree
- No cut corners.
- Ports/packages works like they always do.

- If you can do it with FreeBSD, you can do it with NanoBSD.

NanoBSD features

- Everything is read-only at run-time.
- Safe to pull power-plug.
  - No fsck necessary.
- No missing functionality
  - Unless you remove it yourself.
- Easy to build and customize.

NanoBSD recipe

- `diskless(8)`
  - Gives boot time configurable R/O runtime.
- `Boot0(8)`
  - Choice of which code image to boot.
- One shell script with light magic
  - `src/tools/tools/nanobsd.sh`
- A few convenience features.
  - At no extra cost!

How to build a NanoBSD image

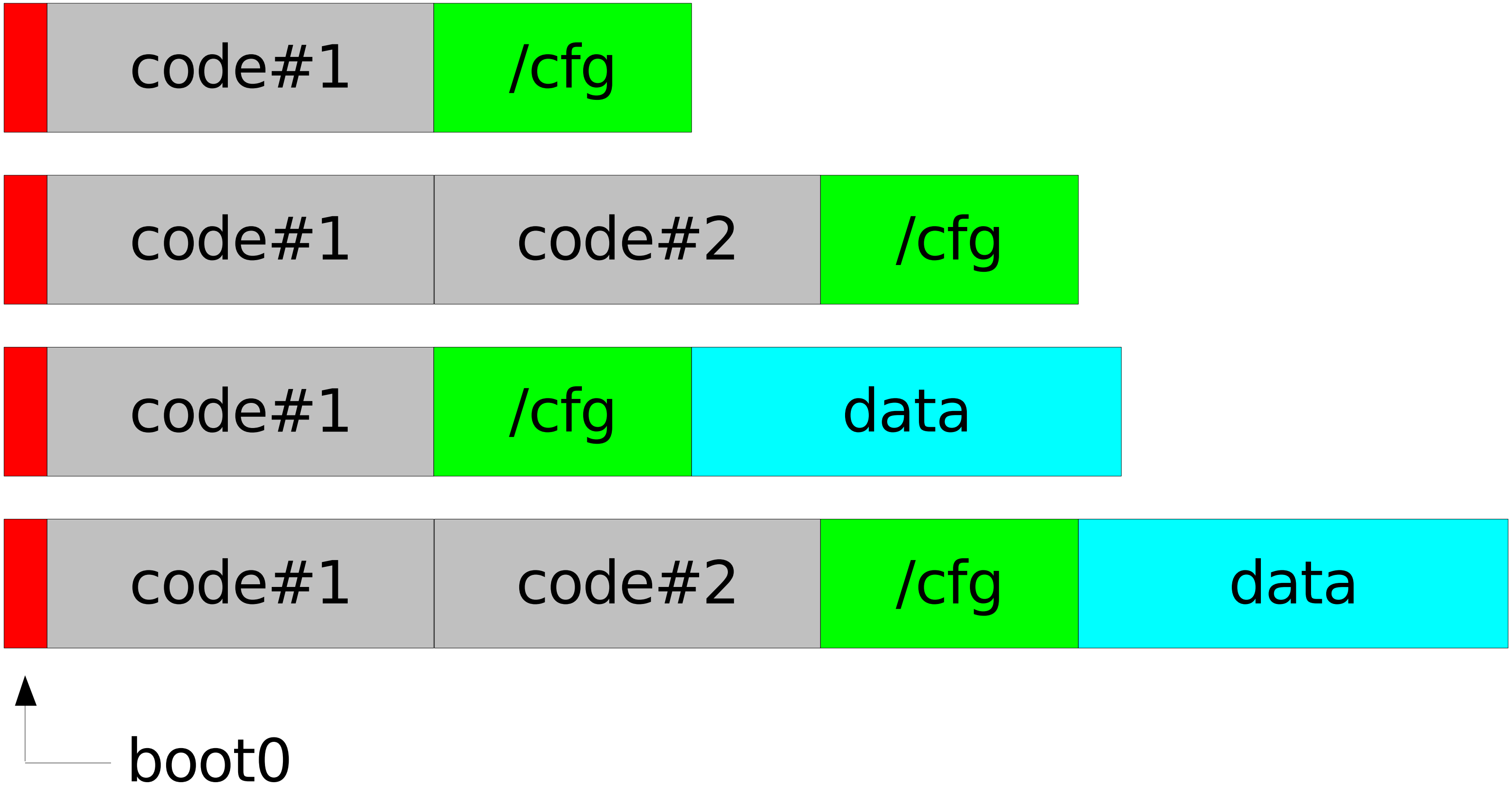
-
  ```
  # cd /usr/src/tools/tools/nanobsd
  # sh nanobsd.sh
  # cd /usr/obj/nanobsd.full
  # dd if=_.disk.full of=/dev/da0 bs=64k
  ```

How to customize Nanobsd

-
  ```
  # sh nanobsd.sh -c myconf.nano
  ```

-
  ```
  # cat myconf.nano
  NANO_NAME=myconf
  CONF_WORLD='
  NO_CXX=YES
  '
  NANO_KERNEL=MYKERNEL
  FlashDevice Sandisk 512M
  #
  ```

NanoBSD disk layouts



/cfg magic

- The config partition contains files for `/etc`
- Partition briefly mounted r/o during boot.
- Remember to save files when you edit:

  ```
  # vi /etc/resolv.conf
  [...]
  # mount /cfg
  # cp /etc/resolv.conf /cfg
  # umount /cfg
  ```

Configuring the media size

- NANO_MEDIASIZE=1048576
  - Count of sectors.
  - Diskinfo(8) is useful.
- NANO_SECTS=32
  NANO_HEADS=16
  - Necessary for some BIOS'es which can't use “packet mode” in boot0(8).

Configuring the media size

- `FlashDevice` *vendor ident*
  - Small library of common device data
  - See `.../nanobsd/FlashDevice.sub`

-
    ```
    FlashDevice SanDisk 1G
    FlashDevice Soekris NET4526
    ```
    etc.

Controlling Media Layout

- `NANO_IMAGES={1,2}`
- `NANO_CODESIZE={0,`*`sectors`*`}`
- `NANO_CONFSIZE={`*`sectors`*`}`
- `NANO_DATASIZE={0,`*`sectors`*`}`

- Zero means “autosize”
- Explicit sizing is more future-proof.

Build/Install/World options

- `CONF_BUILD='...'`
  - Passed to buildworld
- `CONF_INSTALL='...'`
  - Passed to installworld
- `CONF_WORLD='...'`
  - Both `buildworld` & `installworld`.

Customizing

- List of customizing commands.
  - NB: commands without arguments!
- Use shell functions:

```
cust_foo () (
        echo “bar=topless” > \
             ${NANO_WORLDDIR}/etc/foo
)
customize_cmd cust_foo
```

Size of RAM disks

- `/etc` and `/var` are `md(4)` [malloc] disks.
- Default size: 5MB
- Change size:

```
NANO_RAM_ETCSIZE=20480

NANO_RAM_TMPVARSIZE=40960
```

Default customize functions

- `customize_cmd cust_comconsole`
  - Serial console, no gettys on VGA (`/dev/ttyv*`)
- `customize_cmd cust_allow_ssh_root`
  - Allow root to login with `ssh(1)`
- `customize_cmd cust_install_files`
  - Installs files from `.../nanobsd/Files`
  - Contains sysadm convenience scripts

Sysadm on Nanobsd

- `change_password`
  - Changes roots password, saves on `/cfg`
- `save_sshkeys`
  - Saves ssh host keys on `/cfg`
- `updatep1`
  `updatep2`
  - Updates codepartition.

Updating software

-
  ```
  Myhost#  nc -l 2222 < _.disk.image
  ```

-
  ```
  # nc myhost 2222 | sh updatep1
  ```

-
  ```
  # ftp myhost
  get _.disk.image “| sh updatep1”
  ```

-
  ```
  # ssh myhost cat _.disk.image.gz |
     zcat | sh updatep1
  ```

Living with NanoBSD

- Prepare for software updates:

```
# tail /etc/rc.local

if [ -f /etc/ntpns ] ; then
    /etc/ntpns -c /etc/ntpns.conf
else
    /sbin/ntpns -c /etc/ntpns.conf
fi
```

Nailing NanoBSD to the wall