

Building a Console Server using FreeBSD 4.7

CAIA Technical Report 030219A

Grenville Armitage
February 19th, 2003

Introduction

This report summarizes the technology I used to provide a cheap lab console server for Swinburne's Centre for Advanced Internet Architectures (CAIA). The idea is lifted directly from Gregory Bond's [FreeBSD.org article on building console servers](#). My only contribution here is to document a specific configuration used at CAIA - a [VIA EPIA ESP5000 "Mainboard"](#) (motherboard), a [Stallion Technologies EasyIO 4 port serial card \(PCI\)](#), and the [conserver](#) software package. This report is for the benefit of anyone curious enough to try and replicate our scheme.

In a nutshell:

- We'd like remote console access to a bunch of machines in our server room
- Commercial console/terminal servers are rather expensive
- FreeBSD supports multi-port serial cards
- Purchase a multi-port serial card
- Install the free 'conserver' software package:
 - Logs the console outputs of devices attached to our serial ports
 - Allows interactive access to these device
- Voila, we have a console server!

Solution

Our solution is largely based on what we had around, rather than any rigorous evaluation of alternatives. It boils down to two key facts. First, [FreeBSD 4.7](#) running on a motherboard with a standard PCI is likely to support the PCI-card version of Stallion Technologies' EasyIO multiport serial cards. Second, [conserver](#) already exists as a binary package on the FreeBSD CDROMs and provides the desired console server functionality on top of the multiport serial card.

The host machine in this case is a VIA Technologies EPIA ESP5000 motherboard (a fanless 500MHz motherboard in the mini-ITX formfactor) assembled into a Clipper Pro case. FreeBSD 4.7 installation involved booting from CDROM (the ESP5000 has no floppy drive) and then completing the install over NFS from our centre's main file server. The EasyIO 4 port card plugs into the motherboard through a right-angle PCI adaptor, allowing the card to sit horizontally in the otherwise cramped confines of the Clipper Pro case.

The FreeBSD 4.7 kernel source includes an `st1` driver for the EasyIO PCI card, but the default kernel needs to be re-compiled. Add the following line to `/sys/i386/conf/GENERIC`:

```
device st1
```

Rebuild the kernel with the usual "config GENERIC && cd ../compile/GENERIC/ && make depend && make && make install"

Reboot the machine and you should see something like this in `/var/log/messages` if the `st1` driver correctly sees your EasyIO card.

```
st10: <EasyIO-PCI> port 0xe000-0xe07f,0xdc00-0xdc7f irq 5 at device 20.0 on pci0
st10: EasyIO-PCI (driver version 2.0.0) unit=0 nrpanels=1 nrports=4
st10: driver is using old-style compatibility shims
```

The next step is to create the device file entries for communicating with each of the EasyIO's four serial ports. In `/dev` run

```
./MAKDEV st10
```

to build devices `/dev/cuaE{0-7}` (devices 0 through 3 are used by the 4 port card, devices 4 through 7 are relevant if you've purchased the EasyIO 8 port card)

The EasyIO 4 port card has four RJ-45 sockets to fit four serial ports into a single PCI card opening. Console/serial ports of the other equipment in our rack are connected using short lengths of regular cat-5 cable and DB-9 (female) to RJ-45 (socket) adaptors. [See the end of this article for pinouts.]

[Conserver](#) is the software package that turns our multi-port FreeBSD machine into a console server. I installed `conserv-com-7.2.2` (as a precompiled FreeBSD 4.7 package from the first CDROM) with

```
pkg_add /cdrom/packages/All/conserv-com-7.2.2.tgz
```

(If you are missing the CDROM, but are connected to the Internet, you can also download direct from the FreeBSD archives with "pkg_add -r conserver-com".)

`conserver` requires some minimal pre-configuration before it can be used. First, a config file identifying which serial ports should be managed and monitored by `conserver`. I created `/usr/local/etc/conserver.cf` with the following lines:

```
LOGDIR=/var/log/consoles
host1:/dev/cuaE0:9600p:&:
host2:/dev/cuaE1:9600p:&:
host3:/dev/cuaE2:9600p:&:
host4:/dev/cuaE3:9600p:&:
%%
allow: caia.swin.edu.au
trusted: 127.0.0.1
```

This configuration tells `conserver` that all four lines are to be monitored and they're all 9600 baud. The consoles are named `host1`, `host2`, `host3`, and `host4`. Their logfiles will be named `/var/log/consoles/{host1,host2,host3,host4}`. (Before starting `conserver`, create the logfiles directory with "mkdir -p /var/log/consoles".)

One of the features of `conserver` is that you can attach to any one of the monitored serial lines from another machine on the network. The "allow" line in `/usr/local/etc/conserver.cf` says remote console access is permitted from hosts in the ".caia.swin.edu.au" domain (when a password is supplied), and the "trusted" line means remote console access is allowed from clients on the localhost *without* passwords.

Rename the file `/usr/local/etc/rc.d/conserver.sh-` (installed by the `pkg_add` operation) to `/usr/local/etc/rc.d/conserver.sh` so that the `conserver` daemon is restarted each time the local host is rebooted. You can also run `/usr/local/etc/rc.d/conserver.sh {start|stop}` directly to start or stop the daemon without rebooting the machine.

Once `conserver` is running you use the new `console` command to attach to any one of `conserver`'s managed ports specified in `conserver.cf`. For example, to attach to the console of the port named "host2" we would run (while on the terminal server itself):

```
console host2
```

You will now find yourself connected to the device attached to serial port `/dev/cuaE1`. Regardless of whether you are attached to a given port using the console command, `conserver` is continuously logging the traffic on the monitored serial ports and dumping it into the associated logfiles. So, for example, if you were to discover machine on port `/dev/cuaE2` was rebooting at random times overnight you could check the logs in `/var/log/consoles/host3` to see a record of what had been happening.

(Although you can run 'console' on a different host, `conserver`'s daemon will demand password authentication and the passwords will be transmitted in clear text between the remote host and the console server host. Thus it is probably better to make users ssh into the console server and run the 'console' command locally.)

Monitoring FreeBSD hosts

The first application of our console server is to monitor four other FreeBSD machines in our server room. These machines run vanilla FreeBSD kernels, which means they support use of the first serial port ("Com1" in DOS parlance) as the console instead of a keyboard/video combination. The following two steps are required on each monitored host:

- Create a `/boot.config` file with the two characters "-h" as the only line.
- Edit the `ttyd0` line in `/etc/ttys` to read
 - `ttyd0 "/usr/libexec/getty std.9600" vt100 on secure`

The first change ensures the boot loader uses the serial port during boot. The second change ensures an instance of `getty` attaches to the console port once FreeBSD is running (thus allowing logins over the console port).

Speed

Our situation required the `conserver` port rates be set to 9600 to match the speed used by the monitored hosts during the early stages of their boot process (a characteristic of the standard FreeBSD boot loader). If we were not interested in terminal access until the monitored hosts were up and running we could just as easily set higher rates (e.g. 57600 baud) in `conserver.cf` (on the terminal server host) and in the `/etc/ttys` lines of each monitored host.

Stallion Serial Cable Preparation

I purchased RJ45-DB9(female) adaptors that arrive unassembled (the RJ45 side is wired, the wire/pin association on the DB9 side are left to the user). Fortunately their wire color code was the same as this table lifted directly from the [FreeBSD.org article on building console servers](#).

Stallion RJ-45 Pin	Colour	Signal	PC DB-9 Female Pin	RS232 Signal
1	Blue	DCD	4	DTR
2	Orange	RTS	8	CTS
3	Black	Chassis Gnd	N/C	
4	Red	TxD	2	RxD
5	Green	RxD	3	TxD
6	Yellow	Signal Gnd	5	Signal Gnd
7	Brown	CTS	7	RTS
8	White	RTS	1	DCD

Conclusion

We have a console server for the price of a new, low-end motherboard, a four port serial card, and four cat5 cables with RJ45-DB9 adaptors. The EPIA ESP5000 motherboard is actually overkill - any slow, retired PCI-bus motherboard that runs FreeBSD 4.7 would be a suitable host.