Using rsh to run a program on a remote Raspberry Pi

Wouldn't it be nice to be able to run a program on another Raspberry Pi on your LAN and to transfer the ouput of that program back into 2.11 BSD? One could for example read data from a remote sensor on a Raspberry Pi Zero or send commands to a IoT device. For this, the command rsh exist in Berkley Unix, see Berkeley r-commands.

The problem is, that modern Linux systems such as Raspbian do not allow the r-commands because of the insecure nature of these commands. Instead, ssh is implemented. But ssh is not available on 2.11 BSD. Nevertheless it is possible to enable rlogin and rsh on a Raspberry Pi. For security reasons this should only be done on a LAN with a proper firewall, and not on any machine on which private data is stored. One could argue that 2.11 BSD as an ancient operating system is anyway insecure, but a Raspberry Pi is a much more powerful and widely used system. The security risks are therefore much higher.

If you come to the conclusion that the security risk is acceptable in your case, you can enable rsh in Raspbian as follows:

```
sudo apt-get install rsh-server
```

Then, create a file called .rhosts in the pi home directory with the following content

```
bsd_ip_address bsd_user_name
```

where *bsd_ip_address* is the ip address of your 2.11 BSD, and *bsd_user_name* is the name of the user on 2.11 BSD, which should have access to the Raspberry Pi. In my case, this is

```
192.168.1.99 rene
```

Set proper permissions for this file (without this step things will not work!

```
chmod 600 .rhosts
```

That's all on the remote Raspberry Pi.

Log in as root to 2.11 BSD and edit /etc/hosts. Add the line

```
remote_ip_address remote_full_name remote_nick_name
```

where *rpi_ip_address* is the ip address of the remote Raspberry Pi and *remote_full_name* and *remote_nick_name* are the respective full and nick names. In my case this line reads

```
192.168.1.103 pizerow.home.lan pizerow
```

Now login as bsd_user_name and type the following to test your installation:

```
rsh remote_nick_name -l pi -n "ls"
```

You should get a directory listing of /home/pi on the remote Raspberry Pi.

To test something more elaborate, type

```
rsh remote_nick_name -l pi -n "cat sys/class/thermal/thermal_zone0/temp"
```

to get the raw actual cpu temperature of the remote Raspberry Pi.

In the directory rtest in this repository you can find a little C program which shows how to use popen to get this information from within a program, and process it appropriately to produce human readable output in 2.11 BSD

```
rene@pdp11:rtest$ rtest
************************************************
* The CPU temperature on the pizerow is 42.2 deg C *
************************************************
rene@pdp11:rtest$
```