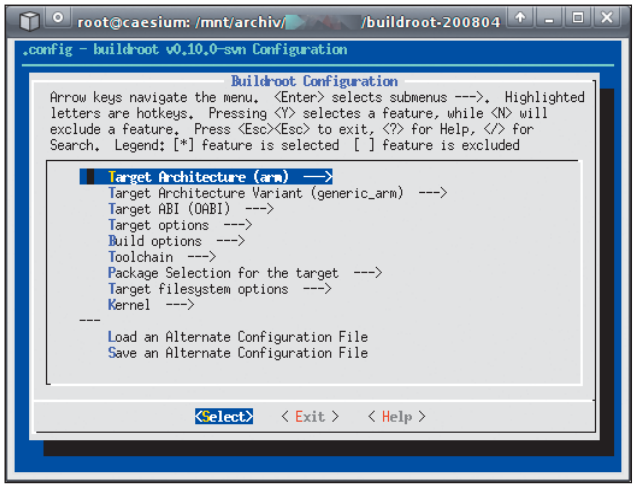
LINUX-HOMESERVER IM MINIFORMAT

NAS-Hacking

Es braucht kein Windows-Home-Server sein – viele Network Attached Storage Devices laufen hinter den Kulissen mit einem abgespeckten Linux. Das lässt sich meist tunen und die Box so um begehrte Server-funktionen nachrüsten. Mit Schwerpunkt auf der ARM basierten NSLU2 zeigen wir Möglichkeiten und Grenzen des Konzeptes.

VON **MATTIAS SCHLENKER**

Dank Festplattenanbindung lassen viele NAS-Geräte die Möglichkeit offen, eigene Startscripte zu definieren, beispielsweise über Debug-Dateien, die beim Systemstart als Shellscript behandelt und ausgeführt werden, gelegentlich über frei zugängliche Startscripte und in einigen Fällen durch System-V konforme Init-Verzeichnisse, in denen Startscripte angelegt werden können. Ist das der Fall, haben Sie meist die Möglichkeit, eine Festplattenpartition mit einem Root-Dateisystem für den verwendeten Prozessortyp und Kernel zu mounten, spezielle Dateisysteme wie */proc* einzuhängen und schließlich mit *chroot* im einen SSH-Daemon oder ein komplettes Startscript im frisch eingehängten Dateisystem zu starten. Mit *chroot* gestartete Prozesse sehen dann das als ersten Parameter übergebene Verzeichnis als Wurzel. So ist es möglich, ein komplett anderes System quasi parallel laufen zu lassen. Als Systeme für den Chroot besonders bewährt haben sich Debian, das für zahlreiche Prozessorarchitekturen verfügbar ist und mit *debootstrap* in ein Verzeichnis installiert werden kann, und *Eriks uClibc Root-Filesystem*. Letzteres lässt sich mit einer Sammlung von Buildskripten leicht selbst erstellen. Was eigentlich für Embedded-Entwickler gedacht ist, hilft bei vielen NAS-Geräten, schnell eine Chroot-Umgebung zur Hand zu haben, die neben den grundlegenden Linux-Tools bereits Serveranwendungen enthalten kann. Dank Verwendung der kompakten C-Bibliothek uClibc bleibt nicht nur der Platz auf Festplatte recht klein, auch der Speicherbedarf gestarteter Anwendungen bleibt gering. Sowohl Debian als auch die uClibc-Root-Umgebung bringen bei Bedarf eine komplette Toolchain mit, also Compiler, Linker und periphere Werkzeuge wie *sed*, *grep*, *awk* und *make* mit.



Mit den Buildroot-Scripten kompilieren Sie eine komplette Root-Umgebung für die Architektur Ihrer NAS-Box – eine native Toolchain erlaubt später die Erweiterung.

Grün oder nicht?

Unsere NSLU2 nahm je nach Last zwischen vier und sechs Watt auf. Ein USB-Festplattenadapter mit 3,5-Zoll-Platte nahm weitere sieben Watt auf. Das liegt weit unter der Leerlaufleistung eines PCs, kommt aber schon nahe an die 30 Watt, die mit Mini-ITX möglich sind. Ab drei Festplatten im 3,5-Zoll-Format sollte daher die Alternative Industrieboard in Betracht gezogen werden.

Zu beachten ist beim Aufsetzen eines Chroots lediglich, dass die Chroot-Umgebung möglichst mit einem Kernel gebaut wurde, dessen Version und Konfiguration sich nicht allzu sehr vom laufenden Kernel unterscheidet: Gelegentlich haben deutliche Altersunterschiede zwischen den Komponenten reproduzierbare Programmabstürze oder hängende Prozesse zur Folge.

Komplettaustausch der Firmware

Ist der Kernel des zu modifizierenden NAS-Systemes zu alt oder fehlen „Einsprungpunkte“, mit denen sich die chroot-Umgebung parallel starten lässt, ist der Austausch der Original-Firmware der einzig sinnvolle Weg. Für den Austausch der Firmware existieren prinzipiell zwei Möglichkeiten: Bei der in der Regel per Script durchgeführten Modifikation der Original-Firmware werden Komponenten, die nicht als Open-Source verfügbar sind, aus einem Originalfirmware-Image in die neue Firmware kopiert. So können meist spezielle Kernelmodule oder das Original-Webinterface erhalten bleiben, während ein SSH-Zugang und ein Paketmanagement nachgerüstet werden. Bei derartigen Scripten existiert oft ein gewisses Risiko, dass bei einer Vielzahl von Optionen nicht alle Original-Komponenten mit den nachgerüsteten Komponenten harmonieren. Schlimmstenfalls ist das Ergebnis ein Gerät, dass nicht mehr per Netzwerk erreichbar und so ein Fall für den Service ist. Die Alternative sind komplett aus freien Komponenten bestehende Images. Die verwenden aber oft sehr einfache Web-Frontends für administrative Aufgaben oder verzichten ganz darauf. *ssh* lautet hier die Devise – nicht nur als Option: Die Secure Shell wird zur Pflicht. Da eine exakte Zusammenstellung zumindest der Basispakete (Bootloader, Kernel, C-Bibliothek, BusyBox und grundlegende Userspace-Tools) gut getestet werden kann, ist das Risiko, das NAS-Gerät zum teuren Briefbeschwerer zu verwandeln, eher gering.

Risiko Bitrot

Dass keine nichttriviale Software fehlerfrei ist, gilt mittlerweile als Binsenweisheit. Zu gefundenen Sicherheitslücken kommt häufig der Wunsch nach Unterstützung auch für neuere Betriebssysteme. Kurzum: Ohne Updates geht es nicht. Wenn Sie eine gering modifizierte Firmware benutzen, die beispielsweise per Chroot auf zusätzliche, auf Festplatte installierte Funktionen zugreift, sind Sie auf verlässliche Firmware-Updates vom Hersteller angewiesen. Greifen Sie dagegen zu einer komplett freien Firmware, sollten Sie sicher stellen, dass das dahinter stehende Projekt aktiv ist und viele Schultern die freie Betriebssystemsoftware tragen. Ein einziger „wohlwollender Diktator“, der andere Entwickler aus „seinem“ Projekt heraushält, mag hinsichtlich der Verlässlichkeit nicht die allererste Wahl sein.

Dass manche Hersteller sich wenig um Sicherheitslücken scheren, solange die Firmware halbwegs funktioniert, zeigt Linksys, deren *NSLU2* zwar als Musterbeispiel für leicht modifizierbare NAS-Geräte gilt, aber das letzte herstellerseitige Upgrade im Herbst 2005 erfahren hat. Mit seither festgestellten Lücken in Kernel und Samba-Server sollte es kein großes Problem sein, die kleine Box mit dieser Firmware zu hacken. Ein positives Beispiel für Upgrade-Politik ist AVM. Für deren FRITZ!Boxen gibt es auch zwei Jahre nach Einstellung noch Firmware, die Sicherheitslücken behebt, und für neuere Boxen existieren „Labor-Versionen“, mit denen AVM Kunden die Möglichkeit gibt, künftige Entwicklungen vorab zu testen. Das macht AVM auch im eigenen Interesse: Als Ziel vieler Modifikationen ist AVM gezwungen, auf „totgeflashte“ Boxen zu reagieren, und möchte mit der „Labor-Firmware“ sowohl Spieltrieb als auch den Wunsch nach mehr Features vieler Nutzer befriedigen.

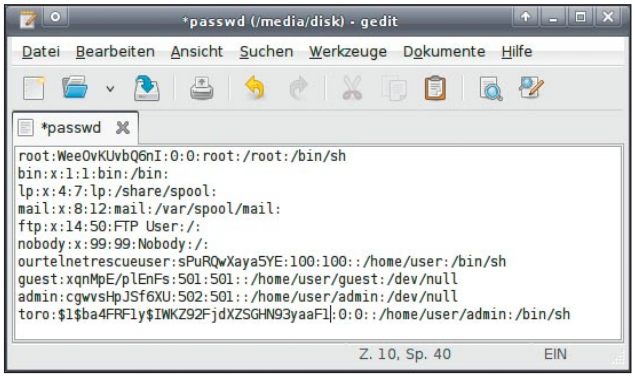
Generell gibt es kein Rezept gegen diese Form des Bitrot. Indikatoren sind jedoch die Größe der Community und der Umgang des Hardwareherstellers mit ihr: Hersteller, die auf die Wünsche freier Entwickler reagieren, eine offene Architektur pflegen oder gar selbst die Open-Source-Community unterstützen, wirtschaften in der Regel nachhaltiger als reine Reseller, die nichtmal ihre BusyBox-Quellcodes offenlegen und nach dem Verkauf nichts mehr vom eigenen Produkt wissen wollen.

Alternative Industrieboard

Jedes NAS-Konzept hat seine Grenzen, die früher oder später erreicht sind. Jenseits dieser Grenzen wird die Wartung aufwändig und der Energiebedarf teuer. Verlockend ist die Erweiterung um immer neue Geräte, schließlich hat das NAS-Device ja ein paar USB-Ports. Reichen die nicht mehr, kommt halt ein USB-Hub zum Einsatz. Dabei ist zu beachten, dass die externen Steckernetzteile vieler Geräte extrem ineffizient sind und hier mal acht, dort mal zwölf Watt zum Stromverbrauch der gesamten Anlage addieren. Da mag es sinn-

Was ist Endianess?

Beim Basteln an Embedded-Systemen wie der NSLU2 oder FRITZ!Boxen werden Sie immer wieder mit den Begriffen *Little Endian* oder *Big Endian* konfrontiert. Diese Begriffe bezeichnen die Anordnung von Zahlen im Speicher. Bei *Big Endian* wird das höchstwertige Byte zuerst gelesen, bei *Little Endian* das niedrigstwertige. Im täglichen Leben arbeiten wir *Big Endian*, die Zahl *eintausendzweihundertundsieben* schreiben wir als 1207. PCs verwenden *Little Endian*, einige Workstation-CPUs *Big Endian*. Viele im Embedded-Bereich anzutreffende CPU-Familien wie PowerPC oder ARM können zwischen beiden Byte-Reihenfolgen umschalten. Das führt gelegentlich zu Verwirrungen: Während die Original-Firmware derNSLU2 im *Big Endian*-Modus läuft (Plattform *armeb*) und hierfür verwendete Chroot-Umgebungen folglich ebenfalls *Big Endian* sein müssen, nutzt die von uns aufgespielte Debian-Firmware den *Little Endian*-Modus (Plattform *arm* ohne Zusatz).



Die Passwort-Datei der NSLU2 können Sie bequem am PC editieren: Verwenden Sie *openssl passwd* um die Hashes zu erzeugen. Hier haben wir einen weiteren Nutzer *toro* mit *UserID 0* angelegt.

voller erscheinen, statt NAS-Gehäuse auf Basis eines anspruchslosen Mini-ITX-Mainboards selbst loszulegen. Freilich ist der Einstandspreis zunächst höher: Statt rund 100 Euro für das NAS sind etwa 170 bis 200 Euro für den kompletten Rechner fällig (Board ab 70 EUR, RAM 20 Euro, Gehäuse Miditower 30 Euro, Netzteil 50 Euro). Dafür erhält man allerdings einen Rechner, der auf Standard-x86-Architektur basiert, sich also mit einem normalen Debian oder Ubuntu ausstatten lässt und der Erweiterungsmöglichkeiten in Form interner IDE-, SATA- und PCI-Schnittstellen bietet. Bei Tests konnten wir so aufgebaute Server mit zwei Festplatten in den Bereich von 30 Watt Leistungsaufnahme drücken und damit so manch fertige NAS unterbieten. Keine Angst: Frickelpotenzial bietet auch so ein Server genug – wie wäre es beispielsweise mit einem System, das von CompactFlash-Karte bootet und nachts nach einem Backup die Datenplatten abschaltet, während der per DynDNS erreichbare Webserver aktiv bleibt?

Selbstbau einer Chroot-Umgebung

Ein von uns gerne gewählter Weg zur Chroot-Umgebung sind die Buildroot-Scripte von uClibc.org, die Sie unter **http://buildroot.uclibc.org** herunterladen können. Nach dem Entpacken des Paketes wechseln Sie in das entstandene Verzeichnis *buildroot* und rufen

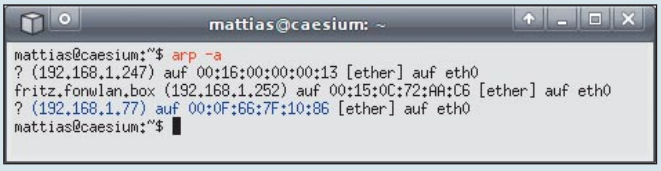
```
make menuconfig
```

auf. Ähnlich der Konfiguration des Linux-Kernels werden Sie mit einem Curses-Interface konfrontiert, in dem Sie Zielplattform, Paketauswahl und viele weitere Optionen bestimmen können. Wählen Sie zunächst die Zielarchitektur entsprechend Ihres NAS-Devices aus. Relativ unkompliziert handzuhaben sind die ursprünglich aus dem PC- und Workstation-Bereich stammenden i386 und PowerPCs, deutlich mehr Optionen haben ARM-Prozessoren und MIPS: Bei diesen können Hardwarehersteller um einen Prozessorkern herum ein maßgeschneidertes Rechenwerk aufbauen. Bei der Identifizierung helfen *uname -a* und *cat /proc/cpuinfo*. Mit *file* geben Binaries und Bibliotheken des laufenden Systems Details preis. Vor dem ersten Durchlauf mit *make* sollten Sie die Paketauswahl fürs Ziel noch zurückhaltend setzen – es geht zunächst darum, eine Crosscompile-Toolchain zu erstellen (also Compiler und Linker zu kompilieren) mit denen Binärdateien fürs Ziel erstellt werden können. Ist das gelungen, passen Sie die Konfiguration an und wählen unter *Package selection for the target* auch *Native toolchain for the target filesystem*. Einen Kernel fürs Ziel benötigen Sie in der Regel nicht. Auch wenn Sie vorhaben, ein x86-Industrieboard von CF-Karte zu booten, ist es meist sinnvoller, den Kernel separat zu erstellen.

Debian auf der NSLU2

Nachdem wir festgestellt hatten, dass die neue auf uClibc 0.9.29 basierende Chroot-Umgebung im Gegensatz zu älteren Versionen nicht für den Praxiseinsatz geeignet war, entschlossen wir uns für die Installation von Debian 4.0r3 auf dem NAS-Gerät – ein Versuch mit Debian 5.0 Beta schlug fehl, und wir mussten die Firmware mit dem Tool *upslug2* flashen. Achtung: Das Firmware-Update mit dem Debian-Kernel kann nur mit etwas Aufwand auf der Kommandozeile rückgängig gemacht werden – Sie sollten sich demnach sicher sein, dass Sie auf ein Webfrontend verzichten können, bevor Sie auf Debian wechseln.

1 Ermitteln Sie mit dem Befehl *arp -a* die Hardware-Adresse Ihrer NSLU2 und tragen Sie diese in Ihre DHCP-Konfiguration ein.



Grund für diese Maßnahme: Der Debian-Installer bezieht seine IP-Adresse dynamisch, die Standard-Firmware nutzt jedoch eine statische IP-Adresse (im Auslieferungszustand *192.168.1.77*).

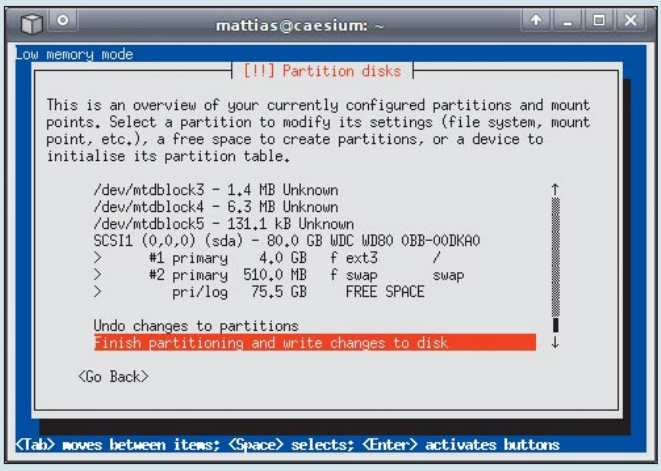
2 Verwenden Sie nicht das originale Debian-Netinstall-Image für die NSLU2 – diesem fehlt die Firmware für den Ethernetcontroller, sie kann also nur mit einem von Debian unterstützten USB-Ethernetadapter verwendet werden. Ein gepatchtes Netinstall-Image finden Sie unter http://www.slug-firmware.net/d-dls.php. Laden Sie das Firmware-Image über die Update-Funktion im Webinterface der NSLU2 hoch.

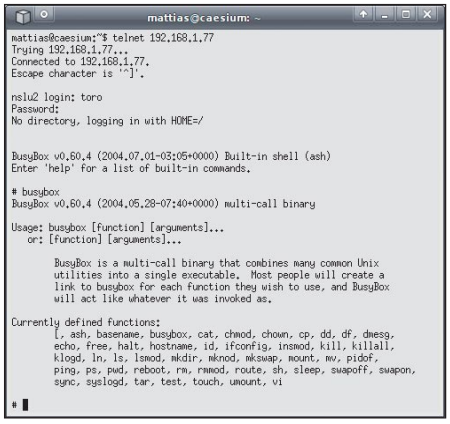
3 Das „Firmware-Upgrade" auf den Debian-Installer und der anschließende Reboot dauern einige Minuten. Angekündigt wird die Bereitschaft zur Installation durch dreimaliges Piepsen. Loggen Sie sich anschließend per SSH mit dem Nutzernamen *installer* und dem Passwort *install* ein, um sich am bekannten Debian-Installer wiederzufinden.

4 Debian schaltet in den *Low Memory Mode*, in dem die Installer-Komponenten selbst ausgewählt werden müssen. Wir wählten *ext3-modules*, *partman-auto*, *partman-ext3* und *usb-storage-modules*.

5 Bei der Partitionierung entschieden wir uns für eine 512 MByte große Swap-Partition und eine vier GByte großen Datenpartition. Den freien Platz werden wir später zuweisen. Die Partitionierung können Sie auch am PC vornehmen, Sie müssen auf der NSLU2 nur noch Mountpoints festlegen.

Die komplette Installation dauert etwa zwei Stunden. Anschließend wird der Installationskernel durch den Arbeitskernel ersetzt und neu gestartet. Sollte die Installation abbrechen, formatieren Sie die Partitionen am PC neu und starten noch einmal. Unter Umständen müssen Sie den gewählten Server wechseln: Bei Betas und nicht offiziell unterstützten Architekturen sind nicht alle Mirror synchron. Sollte die Installation aufwendig zu konfigurierender Pakete wie *texinfo* fehlschlagen, überspringen Sie den Schritt und rekonfigurieren Sie die Pakete nach Abschluss der Installation mit *apt-get -f install* gefolgt von *dpkg-reconfigure -a*.





Telnet-Login in der Originalfirmware: Eine alte BusyBox mit nur wenigen Applets lädt nicht zum Verweilen ein.

WEBLINKS

www.cyrius.com/debian/nslu2
Martin Michlmayr informiert auf seinen Seiten über die Installation von Debian 4.0 und 5.0 auf der *NSLU2*.

www.nslu2-linux.org
Die größte Linux-Community rund um die NSLU2 arbeitet distributionsunabhängig und deckt alle Aspekte von kleinen Hacks der Originalfirmware bis hin zum Komplettaustausch ab.

http://nas-central.org/ALL_COMMUNITIES/Collection_of_NAS-Hacking_communities.html
Sehr umfangreich ist die Übersicht der *NAS Hacking Communities* – hier finden Sie auch Informationen zu Exoten.

www.uclibc.org
Auf den Webseiten den uClibc-Projektes finden Sie Buildroot-Scripte, mit denen Sie Root-Dateisysteme erstellen können.

Gelegentlich bricht die Kompilierung ab, sei es, weil einige Pakete sich nicht für die Zielarchitektur übersetzen lassen oder weil Konfigurationsdateien, beispielsweise für BusyBox, fehlen. Eine BusyBox-Konfiguration können Sie im Hostsystem in einem regulären BusyBox-Quellcode-Verzeichnis mit *make menuconfig* erstellen und dann die erzeugte *.config* ins *buildroot* kopieren – der Suchort wird beim Abbruch angezeigt. Nachträgliche Änderungen an der BusyBox-Konfiguration sind mit

```
make busybox-menuconfig
```

möglich. Mit einer nativen Toolchain und vielen ausgewählten Paketen sollte die Kompilierung auch auf schnellen Rechnern etwa eine Stunde dauern. Je nach ausgewählten Dateisystem-Optionen erhalten Sie dann ein EXT2-Dateisystem-Image oder einen Tarball.

Mit Chroot in die Chroot-Umgebung

Spannend wird der Wechsel in die Chroot-Umgebung. Zunächst ist ein Telnet- oder SSH-Log-in erforderlich, das bei einigen NAS-Geräten bereits geringe Änderungen an der Firmware erfordert. Anschließend ist die Partition zu mounten, auf der Sie das vorbereitete uClibc-RootFS entpackt haben – wenn Ihre NAS-Device ein ext2/3-Dateisystem verwendet, können Sie auch ein Verzeichnis auf einer bereits existenten Partition anlegen. Schließlich müssen noch */proc* und */dev* im Chroot verfügbar gemacht werden:

```
mount -o bind /dev /data/hdd/chroot/dev
mount -t proc none
➥/data/hdd/chroot/proc
```

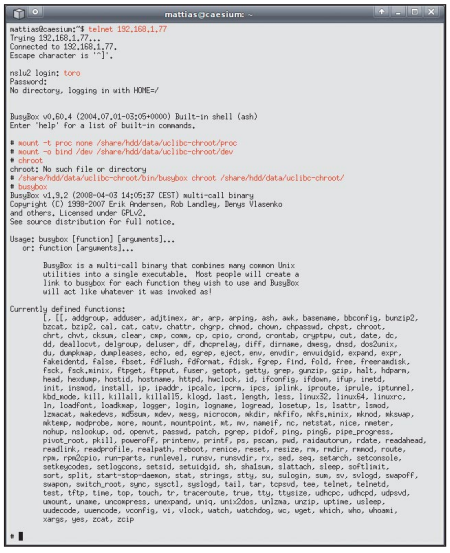
Jetzt erfolgt der Wechsel ins Chroot:

```
chroot /data/hdd/chroot
```

Sollte der BusyBox Ihrer NAS-Device das Applet *chroot* fehlen, hilft es beim Bau des RootFS, einen Durchlauf mit einer statisch gelinkten BusyBox durchzuführen und diese zu sichern, bevor man dann das finale Image mit dynamischer BusyBox baut. Die statische BusyBox kann dann in Situationen verwendet werden, die mehr Applets erfordern als die Original-BusyBox, aber die nicht bereits im Chroot-Käfig durchgeführt werden können. Der Aufruf kann nach Setzen eines Softlinks oder direkt unter Angabe des Applet-Namens erfolgen, beispielsweise:

```
/data/hdd/chroot/bin/busybox.static
➥chroot /data/hdd/chroot
```

Einmal im Chroot wird dann offensichtlich, ob sich der Aufwand gelohnt hat. Im Fall der NSLU2 war dies nicht der Fall: *uClibc 0.9.29* fehlten einige Socket-Funktionen, die in *uClibc 0.9.27* noch vorhanden waren, sodass aus dem Chroot heraus keine Netzwerkzugriffe möglich waren. Anstatt auf Basis alter Root-Dateisysteme weiter zu arbeiten, beschlossen wir einen Wechsel zu Debian.



Ohne Modifikationen an der Originalfirmware erlaubt die Chroot-Umgebung den Aufenthalt in einer Art Betriebssystem-„Parallelwelt“.

Fazit

Viele typische Heimserveraufgaben kann eine modifizierte NAS-Box durchaus übernehmen. Gerade als immer aktive „Medienhalde“, die MP3s und AVIs auch per Stream ins Netz weitergibt, vermeidet die Box, dass ständig ein PC läuft, und hält so Energiekosten gering. Allerdings sollten Sie das Konzept nicht überstrapazieren: Bei sehr prozessorintensiven Aufgaben wie dem Backup per *rsync* oder hohen Ansprüchen an den Datendurchsatz ist das NAS-Modding nicht erste Wahl. **jkn**

Unordnung schaffen

Dauert die Schlüsselerzeugung oder ein SSH-Log-in auf der NSLU2 ewig, liegt das meist an fehlender Entropie: SSH zieht echte Zufallsereignisse heran, um nicht vorhersagbare Schlüssel zu erzeugen. Während aber auf PCs Tastatur- und Maus-Interrupts als Entropiequelle herangezogen werden können, sind Interrupts des Ethernetcontrollers die einzige Entropiequelle der NSLU2. Pingen Sie das Gerät einfach einige Sekunden an, um dringend benötigte Zufallsereignisse zu liefern.