

Using Windows drivers with NdisWrapper

All Wrapped Up

The idea of Linux users scouring the Web for Windows drivers may sound like an April Fool's Day prank, or something from "Candid Camera", but ever since the NdisWrapper Project was launched, this scenario is not as absurd as it might seem.

BY ANDREA MÜLLER



Linux users who are tired of the tangles of wires below their desks, and are thinking of setting up a wireless network to fight the bird's nest, are not exactly spoiled for choice. NICs with native Linux support are few and far between. To be on the safe side, most people opt for an adapter with an Orinoco chipset, like the Lancom Air Lancer MC-11 (see Figure 1). Unfortunately, the only place you can buy these somewhat ancient NICs is at an online auction site like Ebay.

If you would prefer something quicker, like a card that supports the 802.11g standard, models that use the Prism Duette ISL3890 chipset are a good choice, for example, the ALL0271 by Allnet (PCI), or the Netgear WG511GR (PCMCIA). Kernel version 2.6.5 or newer will have a driver by default, but you can convince older Linux versions to cooperate with these adapters if you are prepared to work a little [1].

In many cases the WLAN adapter pre-dates Linux, or it arrives with another piece of hardware. Laptops with the Centrino chip are just one example of this.

To be able to sport the Centrino badge, a laptop needs to have the Intel Pro/Wireless 2100 chip. Some cards that do not have native support can benefit from NdisWrapper [2], and you may be able to get them to talk on Linux.

The Roundabout Way

The NdisWrapper Project is a fairly recent invention, but thanks to the rapid pace of development, it has already reached version 0.9. NdisWrapper is a kernel module that does not support any hardware itself. Instead it emulates an NDIS environment on Linux, making Windows drivers feel at home. NDIS is the abbreviation for Network Driver Interface Specification [3], a specification that was jointly developed by Microsoft and 3COM, which is used by more or less any network interface drivers on Windows.

NDIS is the interface that sits between the second and third layers of the OSI Reference Model [4] as specified by the International Standards Organization (ISO). On one side, it defines a software interface that individual transport proto-

cols can use to communicate with the network adapter, with the network card driver residing on the other side. The NdisWrapper module emulates this interface, allowing the Windows drivers provided with state of the art WLAN adapters to work under Linux. There is a list of supported chipsets at [5].

While I was researching this article, I found out that an entry in the compatibility list does not guarantee that a NIC will work. For example, no matter how hard I tried, I was unable to get a Belkin PCI card with a Broadcom chipset, labeled F5D6001 (printed on top of the bar code) running. I had slightly more luck with a Belkin 802.11g PCMCIA card (product ID F5D7010), which also had a Broadcom and worked perfectly with NdisWrapper 0.8-rc2.

In contrast, I was positively surprised by the Intel-Pro/Wireless-2100 chipsets, and the Netgear MA521, which has a Realtek chipset (see Figure 2). These cards proved to be perfectly stable over a period of several hours using multiple kernel (2.4.22, 2.4.23, 2.4.26, 2.6.3, 2.6.5) and NdisWrapper versions (0.4.0,



Figure 1: The Lancom Air Lancer MC-11 has a Lucent Orinoco chipset. The antenna bulge on the left is characteristic for the card, and includes a connector for external antennas.

0.5.0, 0.6.0, 0.7.0, 0.8, 0.9), achieving transfer rates of between 650 and 830 KBytes per second.

I can recommend Centrino notebooks with the Intel Pro/Wireless 2100 to Linux users; at least you know exactly what chipset you are buying. Also, Intel has transferred a driver to a Sourceforge project [6], thus providing another alternative to using the cards on Linux.

If you are planning on purchasing a WLAN card, because someone you know has the same model running with the NdisWrapper, make sure that the seller will let you return the card and refund the purchasing price if the card doesn't work the way you planned. In the worst case, the manufacturer may have changed the chipset, but kept the name, leaving you with a card that NdisWrapper does not support.

Selecting a Version

Users wanting to run a wireless card with Windows drivers should opt for the latest version of NdisWrapper. Most distributions will supply an older version by default, so check out for people offering packages for your distro. For example, the NdisWrapper homepage has links to RPM packages for Red Hat Linux. Occasionally, you see postings in newsgroups such as *alt.linux.suse* from users, who have created NdisWrapper packages.

Users of Suse Linux 9.1 do not have to go to this trouble, as this version of Suse includes NdisWrapper 0.6 by default. Mandrake Linux 10 also has NdisWrapper, although this is the older version 0.4 which does not support a large number of chipsets. The Intel Pro/Wireless 2100

is quite stable on this 0.4 version.

The jump from version 0.4 to 0.5 was accompanied by a simplified configuration. The source code includes a script that installs, lists, and removes Windows drivers, and stores the module configuration in */etc/modules.conf*. Before an update, Mandrake Linux users will need to remove the older NdisWrapper module first:

```
rm `find /lib/modules/$2
(uname -r) -name`
ndiswrapper.ko`
```

After fulfilling all the system requirements, it is quite easy to add a new NdisWrapper version to the system.

System Requirements

Whether or not a WLAN card will work with NdisWrapper depends on a number of factors:

- the chipset,
- the kernel version – 2.4.20 or newer,
- ACPI.

The easiest way to identify a chipset is to type *lspci -n > before.txt* (before inserting the card) and *lspci -n > after.txt* (after inserting the card). *lspci -n* shows the manufacturer and device ID for a machine's hardware. You can then *diff before.txt after.txt* to view the IDs for your WLAN card. Another approach would be to check the fourth column of the *lspci* output for a combination of IDs that occurs in the *PCI-Ids* (*lspci -n*) in the list of supported chipsets [5].

After fulfilling these conditions, you need to look to the kernel version and ACPI (Advanced Configuration and Power Interface) [7]. Any more or less recent distribution will have the kernel 2.4.20 or later, removing the need to re-compile, as was the case with earlier distributions. NdisWrapper versions prior to and including 0.6 need at least a kernel 2.4.23. This is a requirement that only Suse Linux 9.1, Mandrake Linux 10 or Fedora Core 2 fulfill at present. But users have put pressure on the NdisWrapper developers to modify version 0.7 allowing it to run with older dis-

tributions that use kernel 2.4.20 or newer. Although you can build older NdisWrapper versions with kernel 2.4.22, the modules will not load when you have completed them.

Troubleshooting ACPI

The Advanced Configuration and Power Interface will typically be the toughest nut to crack, especially for laptop users. If you stipulate *acpi = off* or *acpi = ht* for the kernel when booting, that is, if you disable ACPI, loading the NdisWrapper module will make the system unstable, and may even cause kernel panic. *acpi = ht* does not completely disable ACPI, but instead loads just enough ACPI code to support hyperthreading on Pentium IV computers, for example.

Some distributions (Mandrake Linux is one of them) automatically set the boot option *acpi = ht* for a default laptop installation. This is just to be cautious, as some modern devices will not run at all if ACPI is disabled. This said, Mandrake does not check whether the laptop is one of the models affected by this issue. It won't hurt to specify *acpi = force* at the boot prompt, and test whether your machine is stable with power management enabled.

If the test is successful, you need to change the option in the boot manager configuration file. This is */boot/grub/menu.lst* for grub; if you have lilo, you need to edit */etc/lilo.conf* instead, and then call */sbin/lilo* to enable the changes. You simply need to remove the entry for *acpi = off*, or *acpi = ht*, in the configuration file. *acpi = force* at the boot prompt is only needed to overrule any options that disable ACPI. Linux will



Figure 2: The Netgear MA521 has a selling price of about 30 Euro at present, and supports Linux via NdisWrapper.

```

Jun  4 09:05:29 client kernel: ndiswrapper version 0.8-rc2 loaded
Jun  4 09:05:29 client kernel: ndiswrapper adding rtl8180.sys
Jun  4 09:05:29 client kernel: ndiswrapper adding bcmwl5.sys
Jun  4 09:05:30 client kernel: PCI: Setting latency timer of device 02:00.0 to 64
Jun  4 09:05:30 client kernel: wlan0: ndiswrapper ethernet device 00:30:bd:f6:41:20 using driver bcmwl5.sys
Jun  4 09:05:30 client /etc/hotplug/net.agent: register event not handled
/var/log/syslog lines 2139-2144/2144 100%

```

Figure 3: In the second to last line of this output, NdisWrapper indicates that the Broadcom driver has initialized successfully.

automatically use ACPI if no options are specified.

If your machine does not support ACPI, you can still try a workaround with a recent kernel (2.6 if possible), or build a kernel without ACPI support. To do so, you need to disable the *ACPI Support* below *General Setup* in the configuration interface, and rebuild the kernel. To use your distributor's configuration, which will have a name such as *config-kernelversion*, copy it from the */boot* directory to *.config* in the directory with the kernel sources. You need to make this one change after calling *make menuconfig* or *make xconfig*. You can then quit the configuration program, and rebuild the kernel [8].

For some laptops that do not support ACPI on account of an invalid DSTD (Differentiated System Description Table, [9]) – the X10 and X05 models by Samsung, to name just two –, you could possibly try a modified ACPI table [10], which the kernel would load from an initial RAM disk. To allow this to happen you need to build a kernel with the DSTD-from-Initrd patch [11]. Users of Suse Linux 9.0 or newer can save themselves the effort; the Suse kernel has this feature by default. All you need to do is add the path to the alternative ACPI table in YaST (*System | Editor for /etc/sysconfig files | System | Kernel | ACPI_DSTD*).

Setting up NdisWrapper

After setting up your system for NdisWrapper, you can quickly compile the kernel module. Assuming that you have installed the kernel sources, typing *make* in the NdisWrapper directory, will build both the kernel module and the *loadndisdriver*, which is later required to load the Windows driver. *make install* copies the *ndiswrapper* module to the *misc* subdirectory below your module directory, and *loadndisdriver* to */sbin*. Additionally, the *ndiswrapper* Perl script is copied to */usr/sbin*. The script installs and deinstalls the Windows driver, creat-

ing an appropriate module configuration in */etc/modules.conf* or in */etc/modprobe.conf*, if you have a 2.6 kernel running on your system.

Looking for Drivers

The chipset compatibility list provided by the NdisWrapper Project at [3] will point you to the right driver. The *Windows Driver(s)* column has links to downloads, helping you avoid time-consuming searches across manufacturer pages. Most drivers are zipped archives and can be unpacked by entering *unzip driver.zip* in your current working directory. Drivers with a *.exe* extension are often self-extracting zip archives and can be unpacked by typing *unzip driver.exe*.

If you are unable to unpack a required driver archive under Linux, the only workaround available is to launch the installation routine on a Windows system. After accepting the license conditions, store the files in a temporary directory. You can use the Windows search tool to find the driver files, setting the last change date to today as a search condition. This said, an external tool such as filemon [12], which provides details on any files you have open, and also has extensive filtering functionality, may provide more reliable results.

After locating the right directory, you need to copy the driver to a different location, as the setup routine will delete the folder if you cancel the install. After moving the driver to a different location, you can cancel the install. This gives you the files without completing the setup, and without modifying the Windows system. NdisWrapper needs the *.inf* and *.sys* files. A helper script called *ndiswrapper* handles the install.

Call the script by typing *ndiswrapper -i driver.inf*. The script parses the *inf* file to locate the driver and configuration information, and copies both to a directory below */etc/ndiswrapper*. The folder is named after the driver file, for example, *Broadcom-Driver* in */etc/ndiswrapper/bcmwl5*. After inserting the card, you can type *ndiswrapper -l* to check if you have the right driver.

The output:

```

bcmwl5 present
net8180

```

includes a list of installed drivers. The driver for the current card is marked as *present* and is initialized by NdisWrapper when the kernel module is loaded.

If *ndiswrapper -l* does not say *present* although you have a driver, you may have picked the wrong one, or the manufacturer may have created an update that does not work with NdisWrapper. You can try the driver supplied with the manufacturer's CD in this case, removing the need to unpack the driver. All the cards we tested had uncompressed drivers on the CD supplied by the manufacturer.

First Contact

To get your wireless card talking to other WLAN devices, you need to enable the kernel module by typing *modprobe ndiswrapper*. This loads the correct Windows driver. A line like the one shown in Figure 3 should appear in the system logfile. The new network interface then appears in the output from *iwconfig* (see Figure 4).

You can configure the card just like any other WLAN device. *iwconfig* will tell you the name of the interface, typically *wlan0*. Working as root, you can type:

```

[andi@client andi]$ /sbin/iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11g  ESSID:"heimnetz"
          Mode:Ad-Hoc  Frequency:2.462GHz  Cell: 0A:67:00:45:01:06
          Bit Rate:54Mb/s   Tx-Power:14 dBm
          RTS thr:2347 B   Fragment thr:2346 B
          Power Management:off
          Link Quality:100/100  Signal level:-57 dBm  Noise level:-256 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:47  Missed beacon:0

[andi@client andi]$

```

Figure 4: After loading the NdisWrapper module, the output from *iwconfig* should show your WLAN card.


```
iwconfig wlan0 mode Ad-Hoc
```

to set up the interface and specify ad-hoc mode. This is the right choice for a direct connection to another machine. Windows refers to this ad-hoc mode as a *Peer-to-Peer Connection*. If you use an Access Point to manage your network, and assign IP addresses to your devices, you should select *Managed* mode. Master mode, the mode in which you can use a WLAN card as an Access Point on your network, is not supported, as Windows drivers do not support this functionality.

To enable an encrypted connection, and this makes sense for security reasons, you need to assign a WEP key to the card. For example, typing

```
iwconfig wlan0 key 9f65129e50c93c389ebd531da8
```

assigns a hexadecimal WEP key of `9f65129e50c93c389ebd531da8` without the separators. Finally,

```
iwconfig wlan0 essid homenet
```

sets the ESSID. You can use *iwconfig* to ensure that the card really is using all the options you specified, before entering *ifconfig wlan0 192.168.1.6 up* to enable the network interface. You will typically need to supply an IP address for ad-hoc mode, but not for managed mode, as most Access Points also have DHCP server functionality to automatically assign an IP to each wireless device. If you want to use an AP, but can not associate with the AP, the NdisWrapper FAQ [13] suggests that you activate ESSID broadcast at the Access Point, then retry.

Suse Linux has an alternative YaST-based approach to setting up the card. Avoid configuring the wired Ethernet card that the Suse configuration tool mistakenly identifies. In this case, you will not be able to access the wireless options, and your card will not work. The right approach is to launch YaST manually, and select *Other (not detected)*

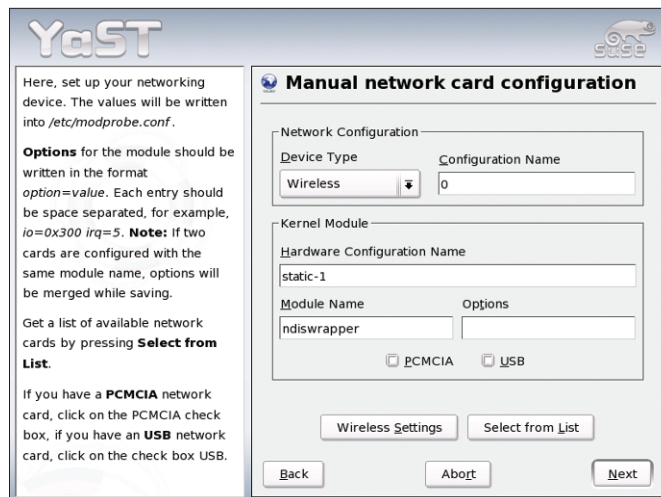


Figure 5: Suse Linux uses YaST to configure wireless network devices. To allow the tool to load the right driver, you need to specify *ndiswrapper* as the module name.

below *Network devices | Network card*. You need to select *Wireless* as the *Device type* and enter *ndiswrapper* in the *Module name* text box (see Figure 5).

Use the *Wireless settings* to define the options such as the mode, the ESSID, and the WEP key. In the case of mobile devices, it makes sense to select *Manual* in *Advanced | Detailed settings | Device Activation* and use the *ifup wlan0* command to enable the wireless interface only when needed. This prevents delays on booting if the device is missing. Internal components typically have a switch to disable the WLAN component, which could help conserve battery power. If you will be using the card permanently, leave the default setting as it is. This tells YaST to enable the WLAN card along with any other network interfaces.

You should not set too much store by the *Link Quality*: display in the output from *iwconfig*. Windows drivers do not have a function for measuring signal quality. The Intel-Pro/Wireless 2100 cards in our test always had a zero in this field, although the card was busy transferring data. The Belkin PCMCIA card with the Broadcom chipset showed values of 97 and 98, and the Netgear card with the Realtek chipset had a full 100. You can check out the statistical values available via the NDIS interface in */proc/net/ndiswrapper/wlan0/stats*.

In contrast to cards with native Linux support, cards with NdisWrapper are not capable of providing a full WLAN solution. Add to that the scary feeling of

risking instability by using a Windows driver to access a hardware device. The willingness of some manufacturers to write free drivers, or at least publish the specifications for their devices, although laudable, will not do much to help a project like NdisWrapper.

On a more positive note, there are innumerable end users who would be unable to use their hardware on Linux without NdisWrapper. Our test demonstrated that NdisWrapper support does not impact the stability or speed, at least for the cards we looked at. Let's not forget

the positive factor of an active developer community, which does not merely focus its activities on bug-hunting, but is genuinely interested in responding to its users' demands for better usability and more chipset support. ■

INFO

- [1] Setting up WLAN cards with the Prism 54 chipset: <http://prism54.org/>
- [2] NdisWrapper Project: <http://ndiswrapper.sourceforge.net/>
- [3] NDIS specification: <http://www.microsoft.com/whdc/device/network/ndis/default.mspx>
- [4] OSI Reference Model explained: <http://www.freessoft.org/CIE/Topics/15.htm>
- [5] List of chipsets supported by NdisWrapper: http://ndiswrapper.sourceforge.net/supported_chipsets.html
- [6] Intel drivers for Centrino WLAN components: <http://ipw2100.sourceforge.net/>
- [7] ACPI: <http://www.acpi.info/>
- [8] Kernel Rebuild Guide: <http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html>
- [9] Timo Hönig, ACPI Implementation: Linux Magazine, Issue 40, March 2004, <http://www.linux-magazine.com/issue/40/ACPI.pdf>
- [10] Overview of Linux ACPI and DSTD: <http://acpi.sourceforge.net/>
- [11] DSTD-Initrd patch: <http://gaugusch.at/kernel.shtml>
- [12] Filemon for Windows: <http://www.sysinternals.com/ntw2k/source/filemon.shtml>
- [13] FAQ for NdisWrapper: <http://ndiswrapper.sourceforge.net/faq.html>