

# Where Linux 802.11x support needs work

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-01/7762.html>

---

*From:* Dan Williams ([dcbw\\_at\\_redhat.com](mailto:dcbw_at_redhat.com))

*Date:* 01/25/05

Date: Tue, 25 Jan 2005 16:47:50 -0500 (EST)  
To: [linux-kernel@vger.kernel.org](mailto:linux-kernel@vger.kernel.org), [netdev@oss.sgi.com](mailto:netdev@oss.sgi.com)

Hi,

This list of stuff that should get fixed in Linux wireless grew out of my attempt to put a GUI on top of Linux wireless with NetworkManager (<http://people.redhat.com/dcbw/NetworkManager>). This isn't, of course, a demand or anything, and I've been personally slowly fixing stuff up as I come to it (orinoco merge, fixing linux-wlan-ng, small kernel wireless driver patches), but I don't think anyone has posted a comprehensive list of where Linux wireless currently falls a bit short.

I think the biggest issue here is that the Wireless Extensions API has stagnated a bit, and driver writers have gone off and done their own thing (for example, WPA support) because the WEAPI hasn't shown leadership in this area. That's fixable, and at this point doesn't seem to be a large amount of work since the main offender here is only WPA.

Second, there are, for historical reasons most likely, areas where the WEAPI has multiple methods of encoding data to/from user space. For example, WE Quality values and WE Frequency/Channel values. Quality is either signed or unsigned 8-bit number, which (I believe) is either a raw dBm/rssi value or a percentage value, respectively. Frequency uses exponent & mantissa notation, OR a channel # stuffed into the exponent/mantissa structure. Things like that.

Comments appreciated, and hopefully this may spark some wider effort to get a few things fixed.

So without further ado, here's the list:

---

- o Quality values vary wildly or are absent
  - 1) atmel doesn't return any quality data from scanned APs
  - 2) ipw\_2100 doesn't return `_any_` quality data (as of v1.02)
  - 3) Different quality methods for almost every driver
    - Prism54 does a quality as a percentage
    - airo mixes use of absolute and relative values in dBm

## Linux-Kernel: Where Linux 802.11x support needs work

- Average and max quality levels for almost all drivers are artificial and don't come from the the card in any way

Work Item: normalize quality values. Wireless extensions supports two different types of quality data, either percentage or dBm. PICK ONE. I would recommend reporting only a Percentage value to user space with the SIOCGIWSTATS call, and having separate ioctl() calls for getting specific dBm/noise values if user-space applications \_and\_ the driver supports it. We cannot have user-space applications guessing which of 3 different quality algorithms the driver is reporting.

- o Frequency values vary wildly from iw\_get\_range
  - 1) prism54 uses completely different exponent values than airo
  - 2) airo, atmel, orinoco are the same

Work Item: Normalize frequency values between wireless cards. Use actual frequencies in MHz rather than using Exponent & Mantissa format as now. Force user-space applications to convert channels->frequencies, based on what frequencies the driver says it supports. Or, fix drivers to report Frequency<->Channel pairs when they report their supported frequencies, but the point again is to PICK ONE and make all drivers do that. Remove the guessing-game from user-space and pick one API for drivers to use.

- o airo/prism54 seem to have problems with ip6 and cause panic
  - 1) Some drivers don't NULL out their data after they are done with it, causing kernel panics later on down the line. See Red Hat bugzilla #135432 for details, Dave Jones has a patch for the airo driver that seems to work better, which is in Red Hat 2.6.10 kernels.

Work Item: Make sure all drivers dispose of and NULL out their data when they close, or fix kernel areas that might depend on that stale data. Or whatever the problem is.

- o Not all drivers have correct netlink support, if they even have it
  - 1) orinoco is too twitchy, sends too many events (shouldn't send them during a scan for example)
  - 2) atmel, airo, and others don't seem to have any netlink support

Work Item: fix all drivers to ensure that when the card successfully associates with an access point, that it signals the kernel that its network link is "up".

- o Not all drivers support wireless scanning
  - 1) orinoco driver mainly, support is upstream and is being slowly merged into the kernel driver

Work Item: Speed up merge of upstream Orinoco into kernel orinoco

## Linux-Kernel: Where Linux 802.11x support needs work

### o Firmware issues

- 1) Cisco aironet firmware upload is quite inconsistent, fails with 5.21 for example. Firmware <= 5.02 seems to be required for using WEP with most access points. Latest Cisco-provided driver is quite different than latest in-kernel driver

Work Item: Figure out licensing issues between Cisco-provided driver for 2.4 kernel (which is MPL) and in-kernel airo driver (which is GPL). Then, figure out what changes were made to the Cisco-provided driver to support firmware up to 5.30.17, and make those changes in the in-kernel airo driver.

### o Ethtool support for all drivers

- 1) viro has done a lot of them, not sure if this is complete.

### o Ad-Hoc mode support is quite flaky or absent from most drivers

- 1) prism54 "mgmt tx queue full" errors on otherwise-working cards
- 2) madwifi resets bitrate to 0 when switching to ad-hoc mode

Work Item: Fix drivers to support Ad-Hoc mode, attempt to get specs on their hardware & registers from manufacturers if we don't have that information yet for all "modern" cards.

### o WPA support is lacking or just in-progress, needs much help

- 1) The point here is that Wireless Extensions API has severely lagged behind the capabilities of current chipsets. There should be support in Wireless Extensions for WPA and its associated technologies, instead of what all the drivers do now, which is separate, non-standard, private ioctl() calls for WPA settings.

Work Item: standardize on an interface for WPA and its associated technologies, and implement that interface in Wireless Extensions API. Fix all drivers to use that API rather than private ioctl() calls. Some drivers that support WPA: atmel, madwifi, prism54, ipw2200. It would also be beneficial in this effort to support the calls that 802.1x stacks need (like wpa\_supplicant and Open 802.1x) so that they don't have to patch the drivers (Open 802.1x) or create special per-driver hook modules (wpa\_supplicant) to be able to capture the necessary authentication packets or set up the card's WPA settings.

### o Drivers deal with hidden ESSIDs differently

- 1) ipw2x00 traps " " and runs of \0 and changes it to "<hidden>" in the driver, while other drivers just pass the blank string through

Work Item: Standardize all drivers to simply pass an empty string through to user-space when the base station does not broadcast its ESSID. Drivers should not be clever about this.

### Levels of Importance (my opinion):

- 1) All drivers MUST support wireless scanning (\*cough\* orinoco \*cough\*)
- 2) WPA support needs to be standardized in Wireless Extensions

## Linux–Kernel: Where Linux 802.11x support needs work

- 3) Consistent (and present) quality data among drivers, both for currently connected AP and for scanned APs
- 4) rtnetlink link notification for all drivers when they associate with an AP
- 5) Ad–Hoc mode support
- 6) Ethtool support
- 7) Cisco firmware issues

–

To unsubscribe from this list: send the line "unsubscribe linux–kernel" in the body of a message to [majordomo@vger.kernel.org](mailto:majordomo@vger.kernel.org)

More majordomo info at <http://vger.kernel.org/majordomo–info.html>

Please read the FAQ at <http://www.tux.org/lkml/>

---

- TEXT/plain attachment: [linux–wireless–problems.txt](#)