WiFi control plane overview

Johannes Martin Berg

2009-02-26
Introduction

We’ll cover

- some background
- wext (and quickly forget about it)
- cfg80211/nl80211
- wpa_supplicant
- hostapd
Introduction – History (non-technical)

1996 or so
? - today
during mac80211 cleanup

late 2006
since then
February 2009

Jean Tourrilhes creates wireless extensions
userspace MLME used by Jouni Malinen
wireless extensions deemed unsuitable for future configuration needs
initial cfg80211/nl80211 work (myself)
many extensions in nl80211, mesh, HT, ...
userspace MLME idea dropped
Architecture – current

userspace

nl80211

cfg80211

cfg80211_ops

mac80211

current
non-mac80211
drivers
Architecture – planned

userspace

nl80211  wext (for backward compatibility)

cfg80211

cfg80211_ops

mac80211 and future fullmac drivers
background

netlink

- RFC 3549
- used to communicate with kernel (in theory also userspace to userspace)
- TLV-based protocol
- implements “families”
- one specific family: generic netlink
- easily extensible, discoverable
Wireless extensions – code structure

- all code is in net/wireless/wext.c
- not much code – drivers need to implement a lot
Wireless extensions – main flows

- userspace sets each parameter one by one
- driver tries to work with these parameters
- problem: is the user going to send a BSSID after the SSID?
Wireless extensions – handoff points

- `netdev.wireless_handlers`
  - contains array of standard and private handlers
  - handlers called by userspace via ioctl
- drivers send events via netlink
Wireless extensions

- all executed under rtnl
- callbacks all run in process context (from userspace)
- event sending can be done in any context
cfg80211

- thin layer between userspace and drivers/mac80211
- mainly sanity checking, protocol translations
- thicker than wext – sanity checking, bookkeeping
cfg80211 – code structure

All files except the header files (include/net/cfg80211.h and include/net/wireless.h) are in net/wireless/.

- Kconfig, Makefile: build system
- core.c, core.h: core code
- nl80211.c: nl80211 generic netlink code
- scan.c: scan code
- reg.c: regulatory enforcement code
- util.c: some utility functions for cfg80211 and drivers
- radiotap.c: a radiotap parser (for injection)
- wext-compat.c: wext compatibility code
- sysfs.c: sysfs representation
cfg80211 – nl80211

- userspace access to cfg80211 functionality
- defined in include/linux/nl80211.h
- currently used in userspace by iw, crda, wpa_supplicant, hostapd
cfg80211 – main flows

- device registration
- regulatory enforcement
- station management (AP)
- key management (AP only)
- mesh management
- virtual interface management
- scanning
cfg80211 – main flows

device registration

- drivers register a struct wiphy with cfg80211
- this includes hardware capabilities like
  - bands and channels
  - bitrates per band
  - HT capabilities
  - supported interface modes
- needs to be done before registering netdevs
- netdev ieee80211_ptr links to registered wiphy
cfg80211 – main flows

regulatory enforcement (overview)

- still work in progress
- relies on userspace helper (crda) to provide restriction information
- will update the list of registered channels and (optionally) notify driver
cfg80211 – main flows

regulatory enforcement

- default: restrictive ’world’ regulatory domain
- driver/user/AP (11d) tells us where we are (iso 3166 code like ’US’)
- create a uevent to notify userspace
- udev runs crda, which parses database and uploads information via nl80211 to kernel
- depending on origin of hint, information may be postprocessed
- channel lists of all wireless devices are updated with regulatory flags
- future channel use will take new restrictions into account, e.g. while scanning
cfg80211 – main flows

station management

- add/remove/modify stations
- dump station list
- works with a few callbacks:
  - .add_station
  - .del_station
  - .change_station
  - .get_station
  - .dump_station (races!)
cfg80211 – main flows

mesh management

- mesh path handling to station handling
- mesh parameters can be set/retrieved
cfg80211 – main flows

virtual interface management

- create/remove virtual interfaces
- change type of virtual interfaces (provides wext handler)
- change 'monitor flags'
- keeps track of interfaces associated with a wireless device
virtual interface basics

- optional
- mostly for mac80211, though other appropriate uses exist
- only matching PHY parameters possible, e.g. all virtual interfaces are on one channel
- driver responsible for rejecting impossible configurations
cfg80211 – main flows

virtual interface types

• ad-hoc (IBSS)
• managed
• AP and AP_VLAN
• WDS
• mesh point
• monitor
  • can set monitor flags: control frames, other BSS frames
  • special case: cooked monitor
  • cooked monitor sees all frames no other virtual interface used
cfg80211 – main flows

virtual interface use

- monitor (replacing things like IPW2200_PROMISCUOUS and module parameter)
- switching modes like with iwconfig
- allow multiple interfaces, combining e.g. WDS and AP for wireless backhaul
cfg80211 – main flows

scan features

• many more features than wext:
  • multiple SSIDs
  • channel specification
  • allows IE insertion

• extensible via generic netlink attributes
cfg80211 – main flows

scan flow

• userspace request (nl80211 or wext)
• handed to .scan handler in a structure specifying what to do
• mac80211/driver scans according to the request
• beacons/probe responses handed to cfg80211 to fill BSS list (cfg80211_inform_bss_frame())
• request struct given back to cfg80211 with indication whether scan was successful or not (cfg80211_scan_done())
• userspace notified via nl80211/wext that scan is done
• userspace requests BSS list
cfg80211 – synchronisation

- global lock held for list/regulatory management
- per-device lock held for callbacks, device data structures
- configuration calls to drivers executed under rtnl
- this synchronises against interface callbacks (start, stop, etc.)
Userspace

most common tools

- NetworkManager
- wpa_supplicant
- hostapd
- "userspace MLME"
Userspace – NetworkManager

- GUI tool for GNOME and KDE
- uses hal/d-bus
- more importantly, uses wpa_supplicant (except for scanning, uses wext)
- therefore no big concern – look at wpa_supplicant instead
Userspace – wpa_supplicant

- internally modular architecture, supports multiple backends
- current git version supports nl80211 scanning
- current git version can try nl80211 and fall back to wext
- nl80211 backend (‘driver’) still uses some wext calls
- actively maintained by Jouni Malinen (Atheros)
Userspace – hostapd

- implements (almost) the entire AP MLME
- works with mac80211 through nl80211
- requires working radiotap packet injection
- requires many of the nl80211 callbacks
- requires ‘cooked’ monitor interfaces
- actively maintained by Jouni Malinen (Atheros)
Userspace – “userspace MLME”

- misleading name for “client-MLME in userspace”
- initially seen as only/easiest way to implement 802.11r (due to auth frame processing)
- hard to do right
  - kernel needs to know assoc status for packet flow
  - quite backwards – userspace creating e.g. HT or rate information based on what device supports and then passing it through
  - software scanning in userspace doesn’t go too well with firmware
  - configuration big problem – wext/nl80211 no longer applicable
Userspace – “userspace MLME”

What to do about it?

- remember wext
  - can set only SSID/BSSID
  - either operation can trigger authentication/association
- discussion with Jouni about further move to nl80211
- initial idea: just associate with parameters
- problem: deauth still needed, but associate vs. deauth and disassoc is asymmetric
Userspace – “userspace MLME”

What to do about it? (2)

- will implement auth/assoc separately
- support multiple authentications simultaneously
- support adding arbitrary IEs into auth/assoc frames
- together this allows 802.11r
- tools need to implement auth/assoc, provide example in iw
- no need to put more of the MLME into userspace
- auth/assoc state machine needed in cfg80211 for wext
- open question: what to do with drivers that don’t support separate auth/assoc?
Stay up-to-date

- subscribe to wiki changes on these pages
- follow patches going in: 
  git log -- net/wireless/ include/linux/nl80211.h
- read the wireless list
  (http://wireless.kernel.org/en/developers/MailingLists)
Thank you for your attention.

Questions?