Analysis of open source drivers for IEEE 802.11 WLANs

Vipin M

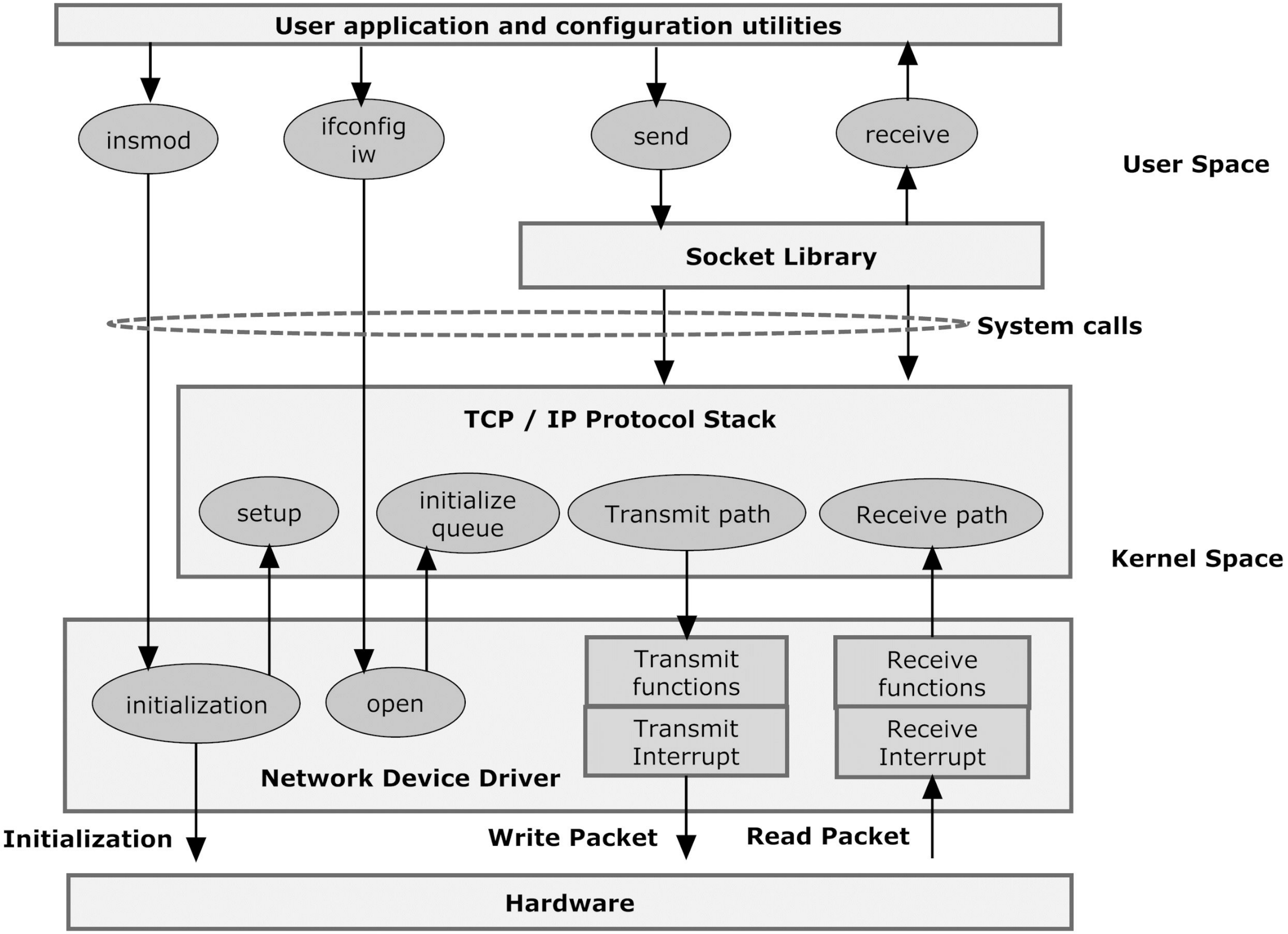
Under the guidance of :

Dr S Srikanth
AU – KBC Research Centre
MIT Campus of Anna University
Chennai

Overview

- Linux Network Driver
- WLAN Driver
- Evolution
- MADWiFi
- Linux Kernel Stack
- Functional blocks and flow of operation
- Control Plain
- Configuration and management path
  - Adding / Deleting an Interface
  - Scanning
  - Authentication and Association
  - Tx Power
- SoftMAC
- Hardware driver
- Special operations
  - Monitor mode
  - debugfs

Linux Network Driver



WLAN Driver

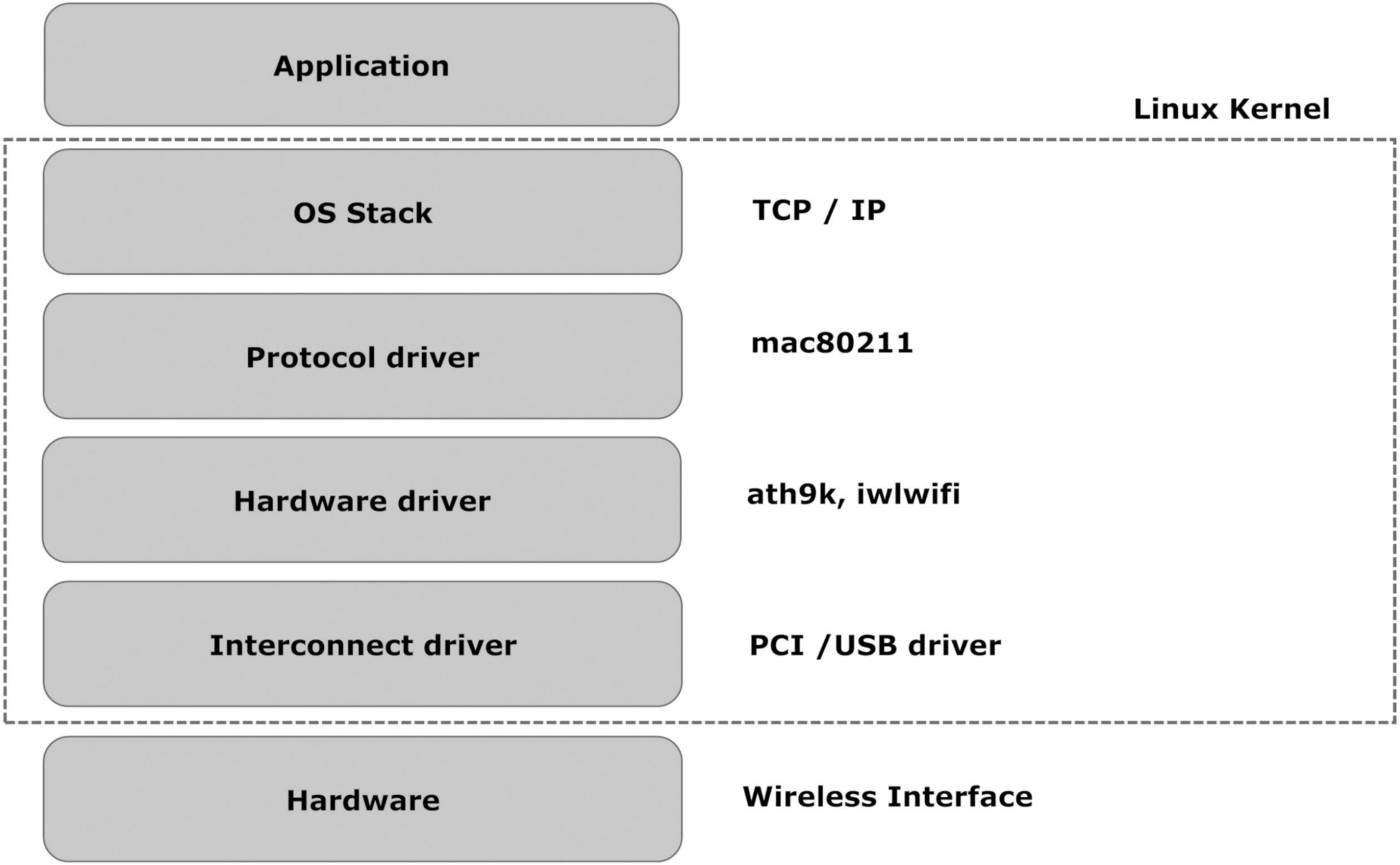
- IEEE 802.11 drivers are like any other network driver
- WLAN drivers support
  - Ad-hoc
  - Infrastructure
  - Mesh
  - WDS (wireless distribution system)
  - VAP (virtual access point)
  - Virtual interface
  - Monitor

Evolution

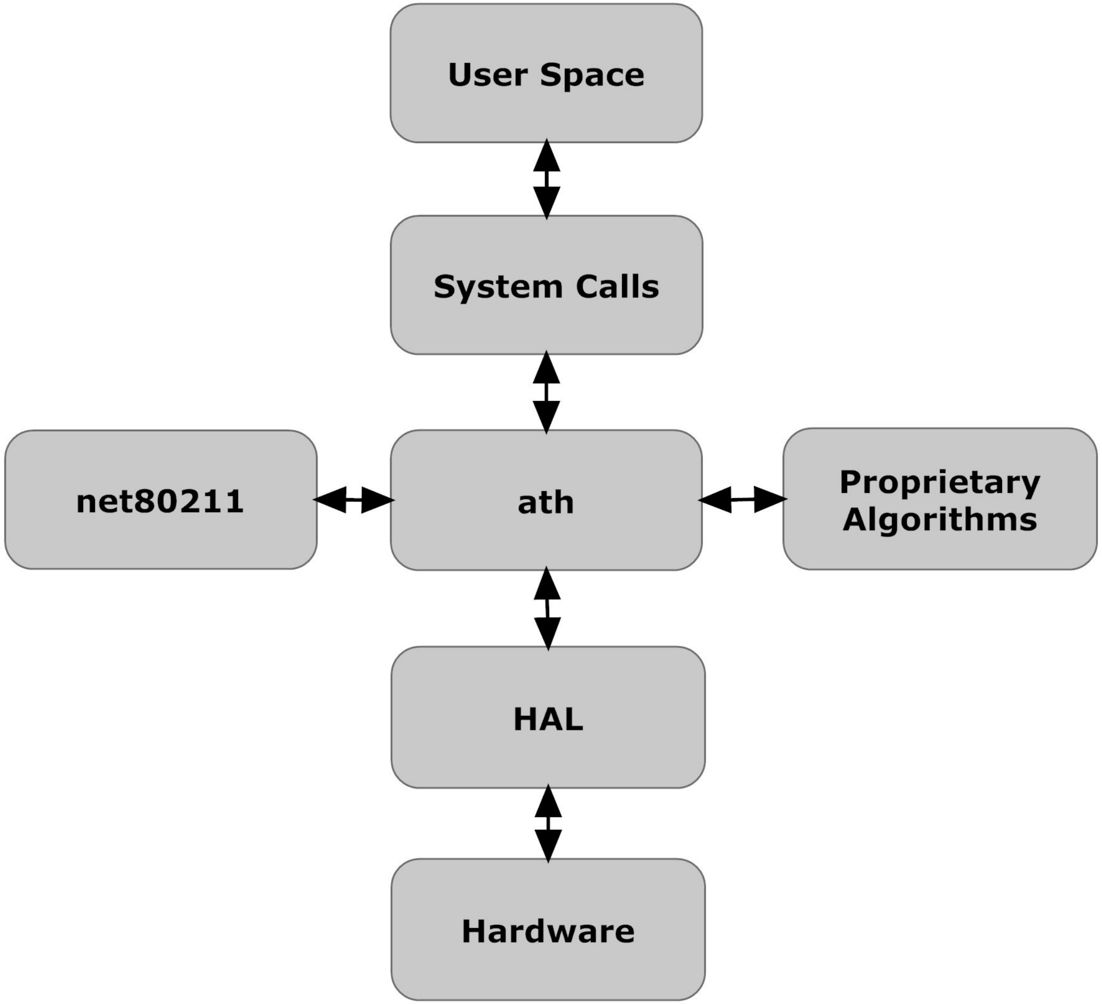
- Implementation
  - Full hardware MAC
  - Partial hardware
  - Full software MAC

- Source code
  - Proprietary
  - Partial open source
  - Fully open source
    - Part of Linux kernel tree

Linux Kernel Stack



MADWiFi

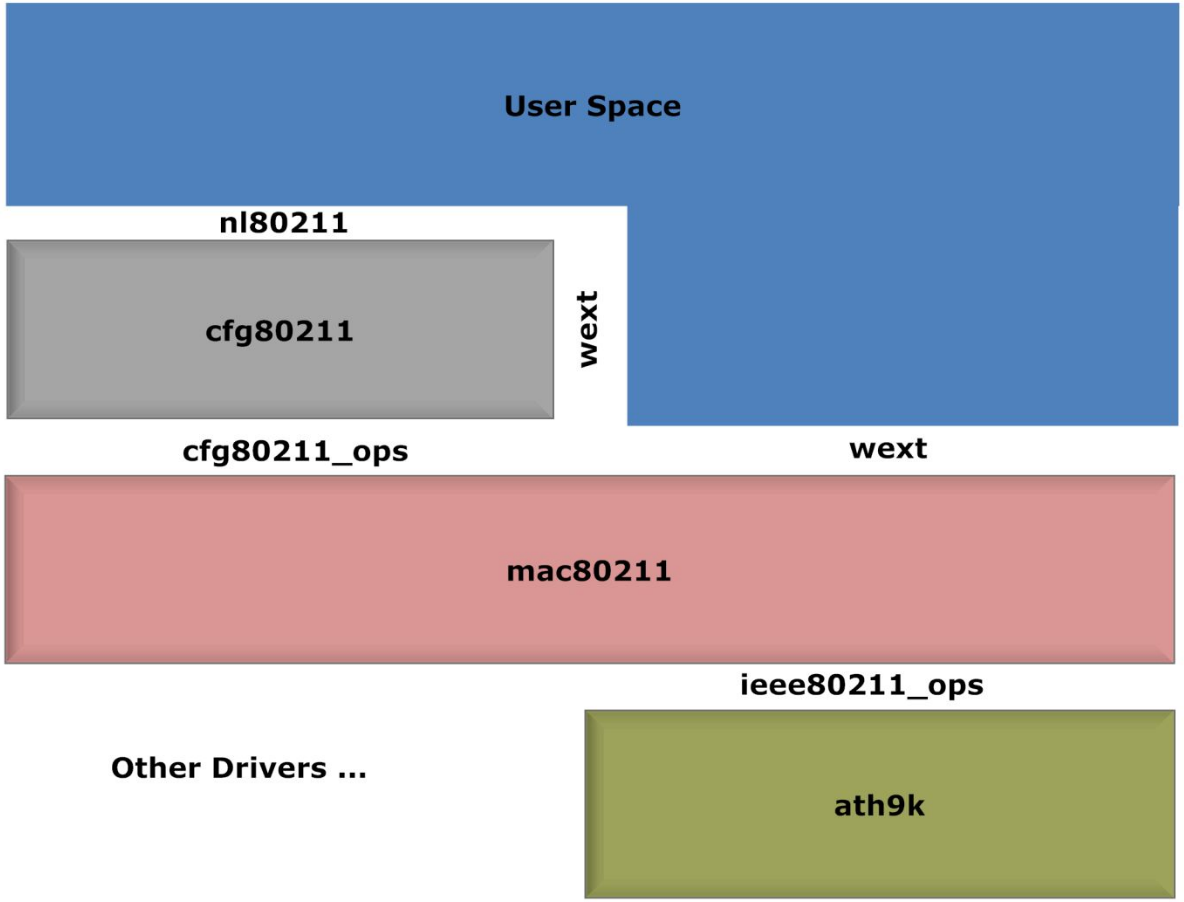

User Space
System Calls
net80211
ath
Proprietary Algorithms
HAL
Hardware

Functional blocks and flow of operation

- Functional blocks
  - Control plane
  - SoftMAC
  - Hardware driver

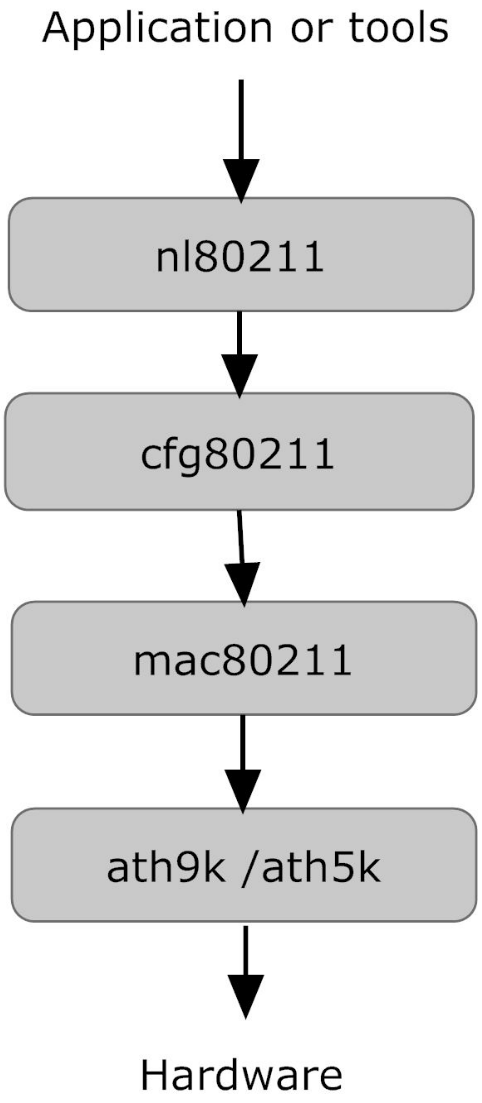
- Flow of operation
  - Configuration and management path
  - Transmit and Receive path
  - Special operations

Control Plain



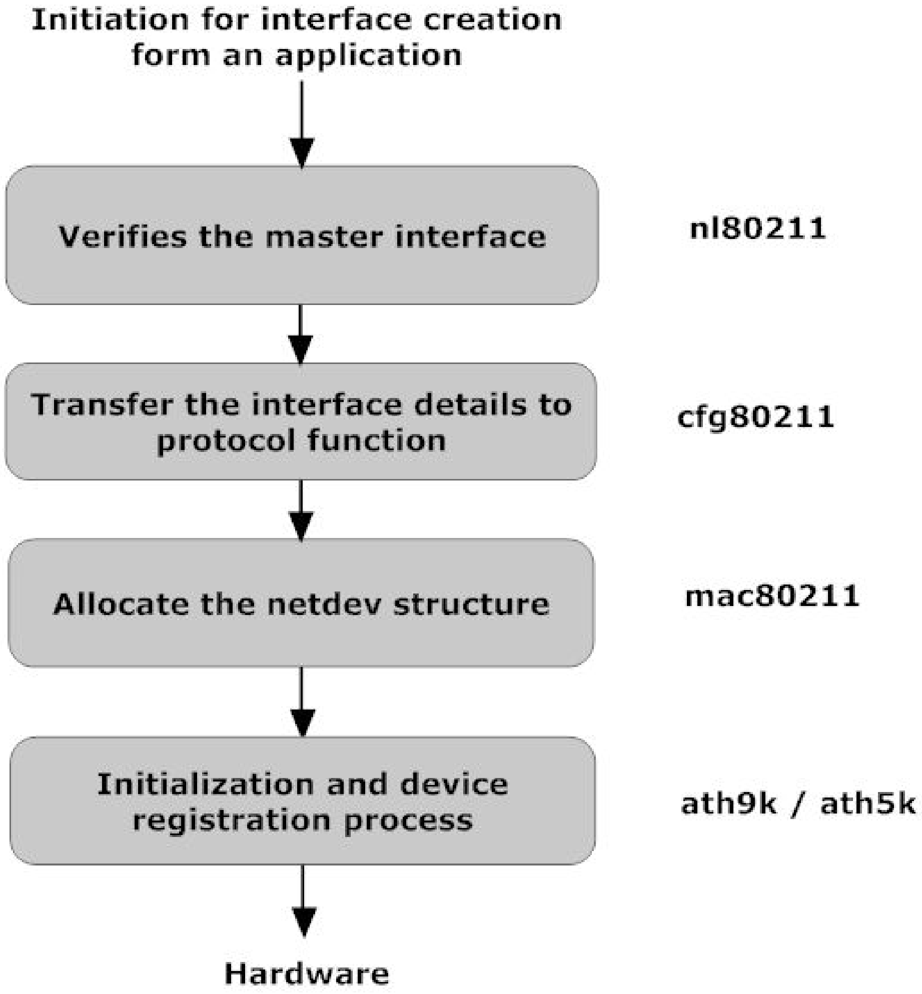
- User Space
- cfg80211

Configuration and management path



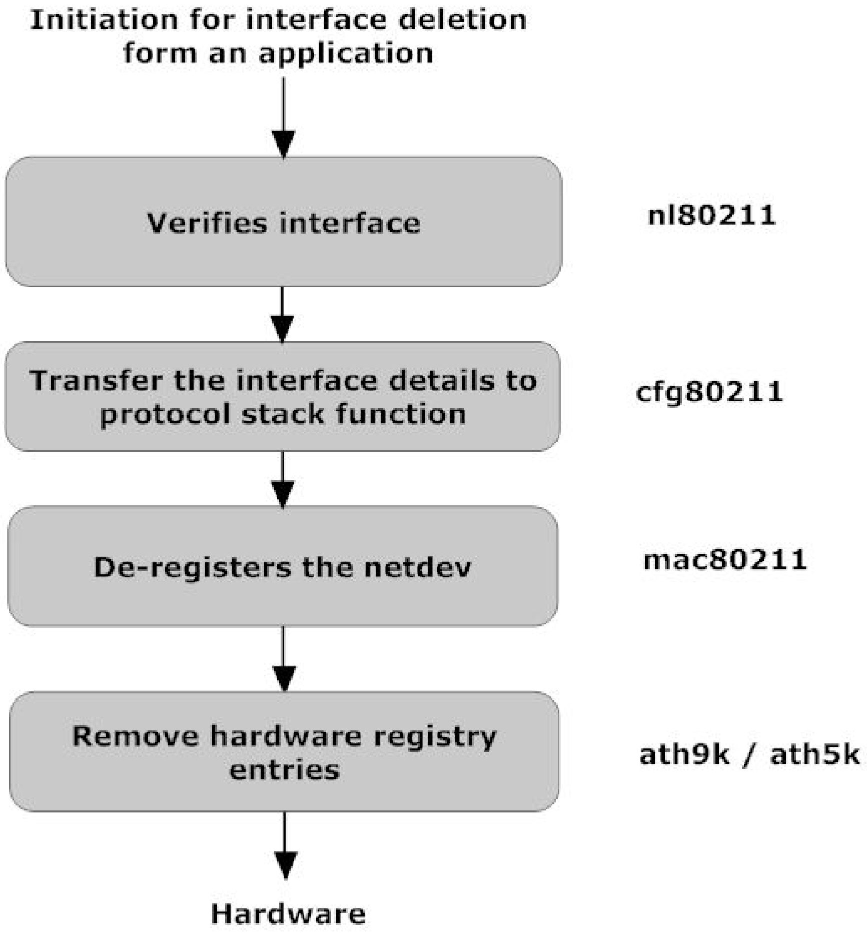
- Initiated from user level
- Each layer act based on the operation

Adding an Interface


Initiation for interface creation form an application
Verifies the master interface
nl80211
Transfer the interface details to protocol function
cfg80211
Allocate the netdev structure
mac80211
Initialization and device registration process
ath9k / ath5k
Hardware

- This is initiated by user level tools to add extra virtual interface.

Deleting an Interface



Initiation for interface deletion form an application

Verifies interface — nl80211

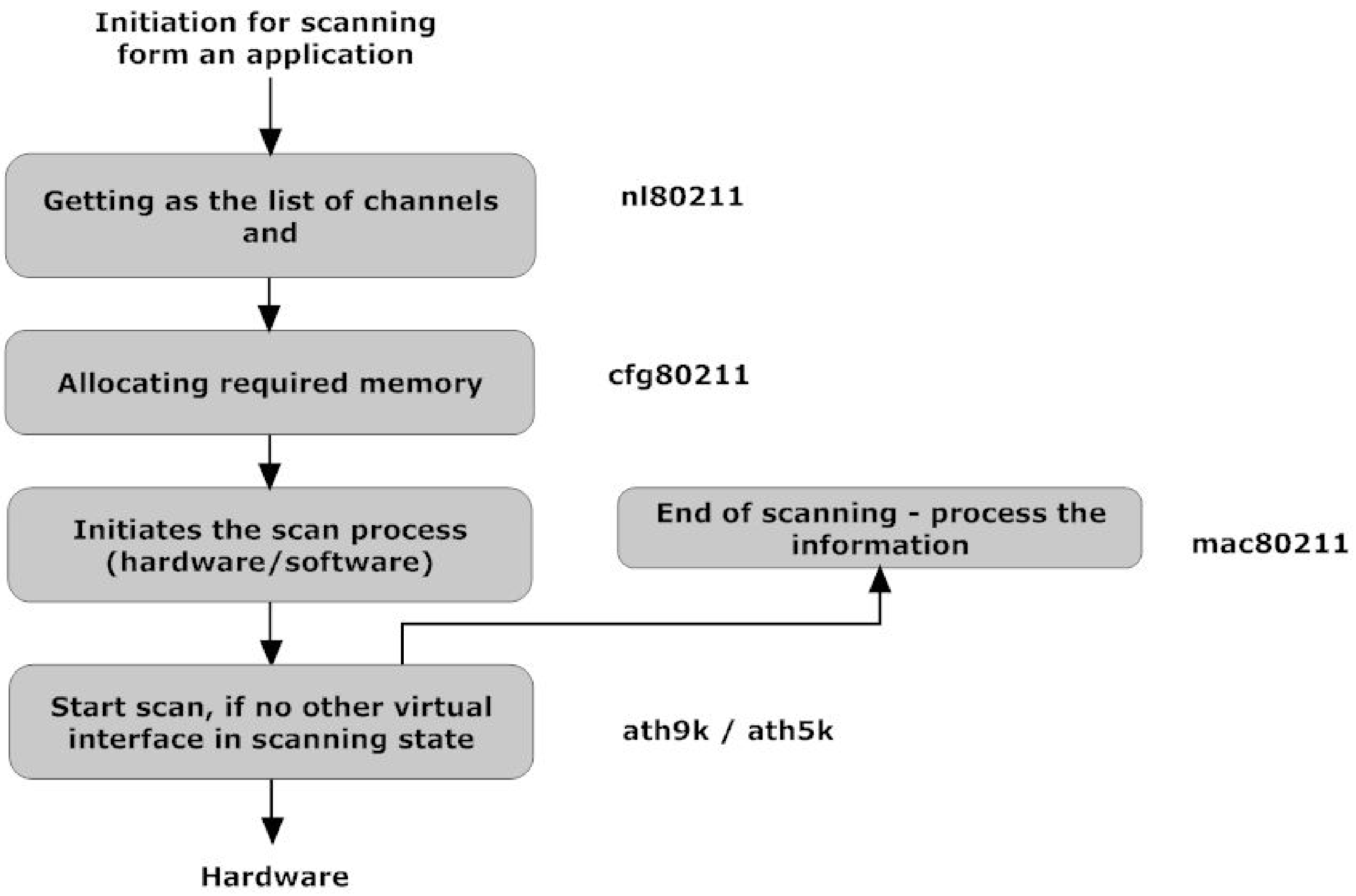
Transfer the interface details to protocol stack function — cfg80211

De-registers the netdev — mac80211

Remove hardware registry entries — ath9k / ath5k

Hardware

Scanning



Initiation for scanning form an application

Getting as the list of channels and

nl80211

Allocating required memory

cfg80211

Initiates the scan process (hardware/software)

End of scanning - process the information

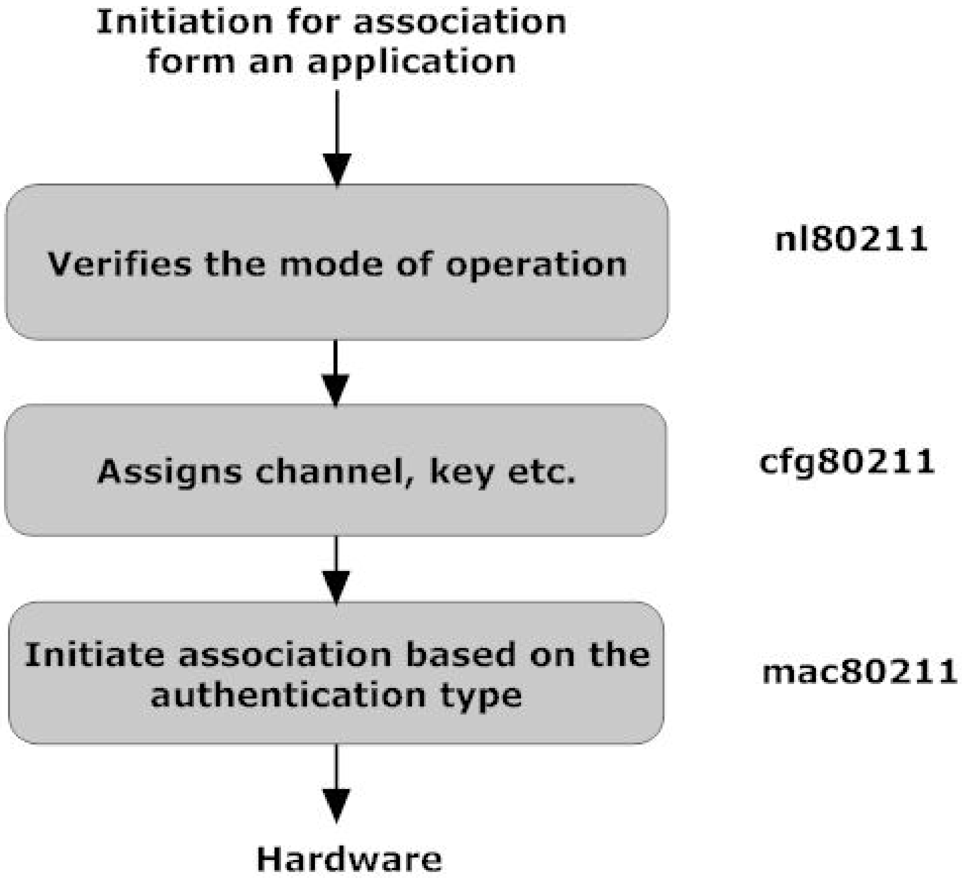
mac80211

Start scan, if no other virtual interface in scanning state

ath9k / ath5k

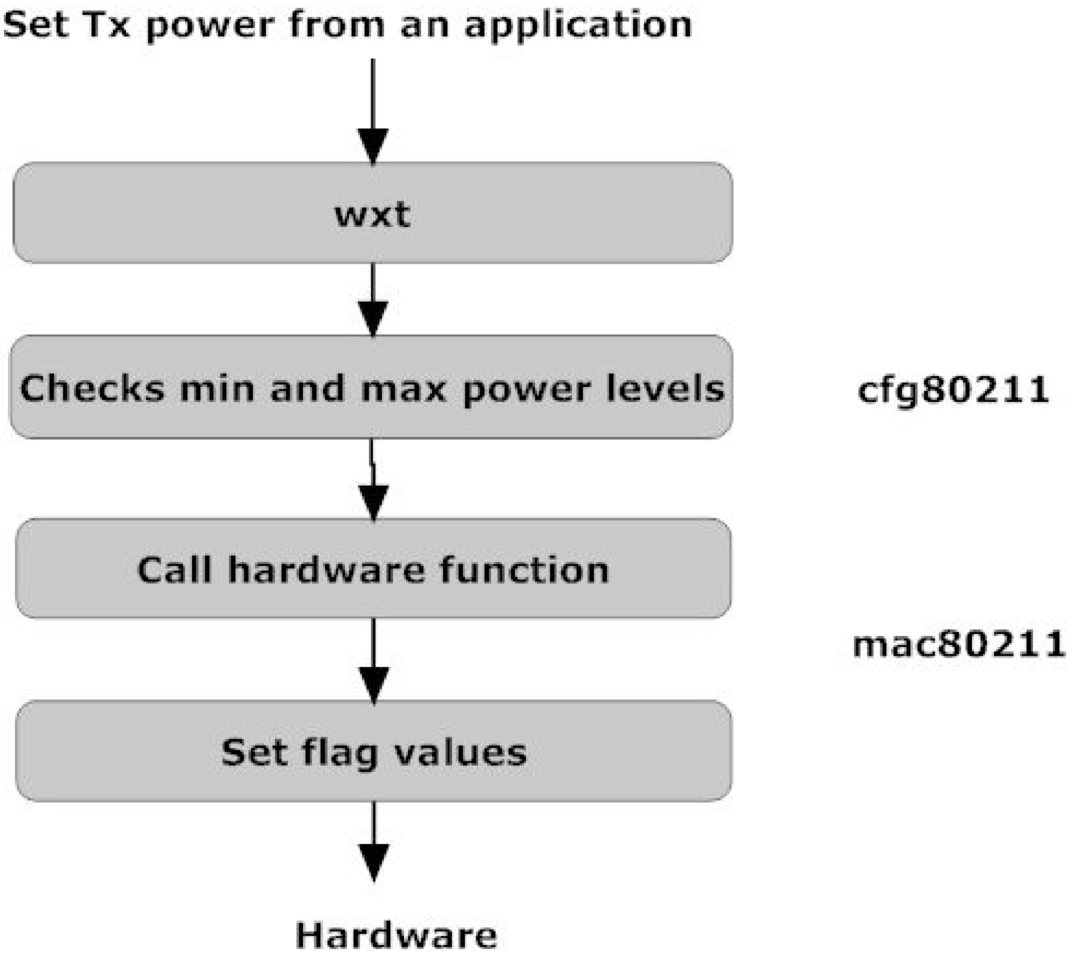
Hardware

Authentication and Association



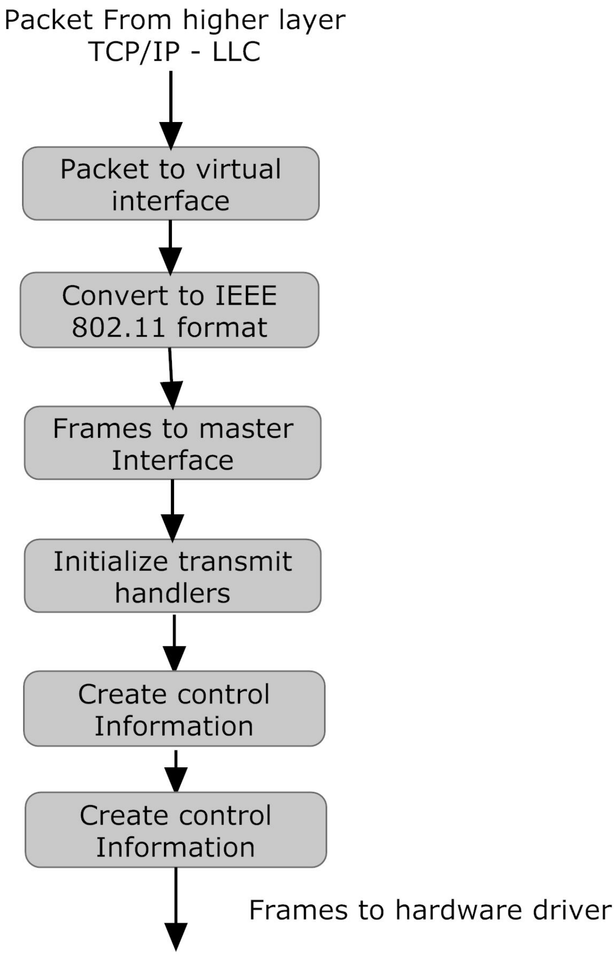
Initiation for association form an application
Verifies the mode of operation
nl80211
Assigns channel, key etc.
cfg80211
Initiate association based on the authentication type
mac80211
Hardware

Transmission Power



SoftMAC

- Transmission Path
- Receive Path



Packet From higher layer
TCP/IP - LLC

Packet to virtual interface

Convert to IEEE 802.11 format

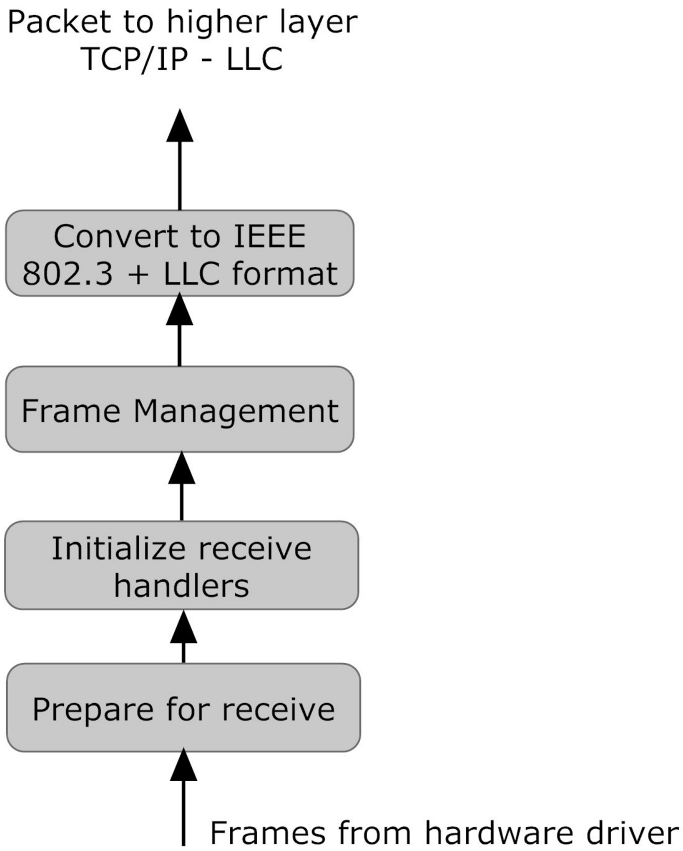
Frames to master Interface

Initialize transmit handlers

Create control Information

Create control Information

Frames to hardware driver



Packet to higher layer
TCP/IP - LLC

Convert to IEEE 802.3 + LLC format

Frame Management

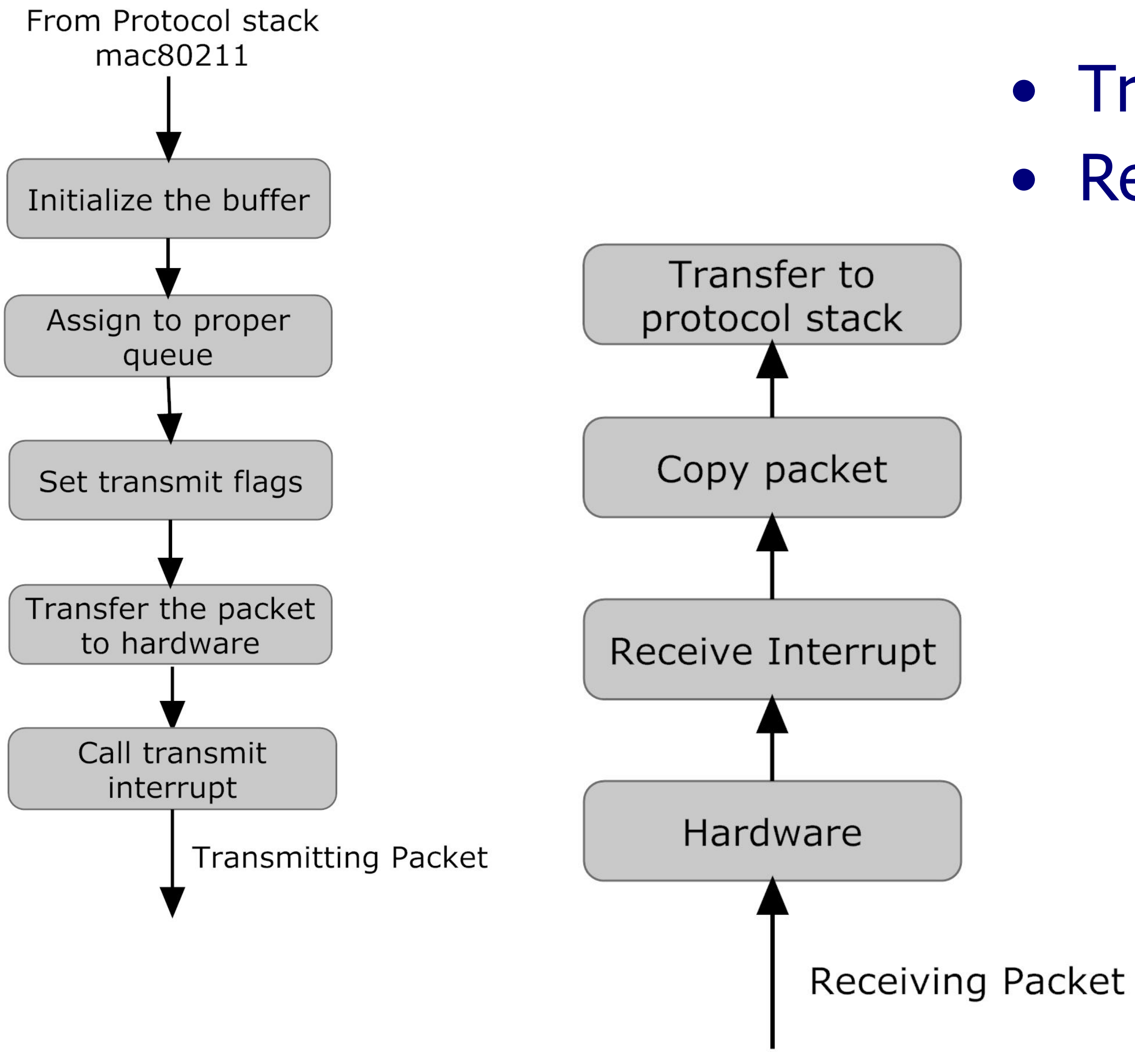
Initialize receive handlers

Prepare for receive

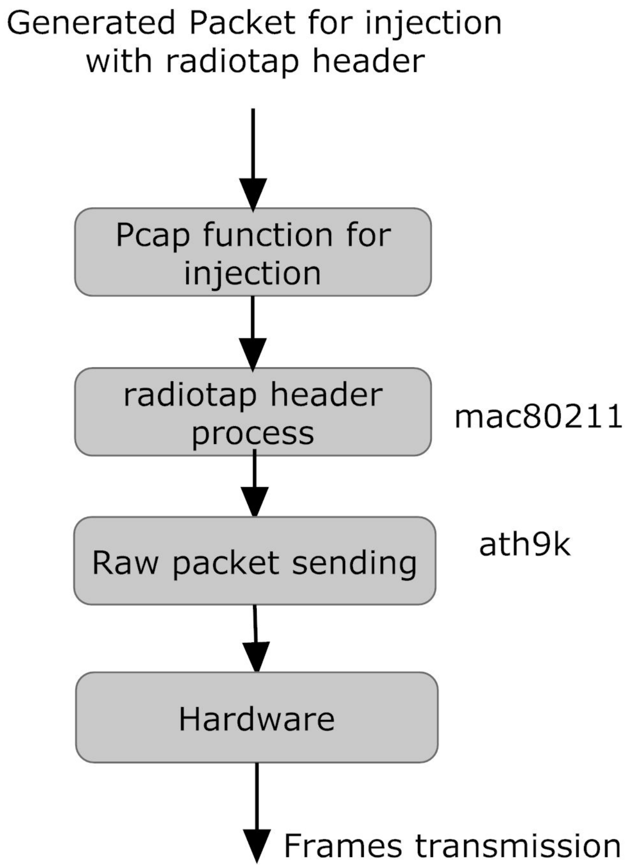
Frames from hardware driver

Hardware driver

- Transmission Path
- Receive Path

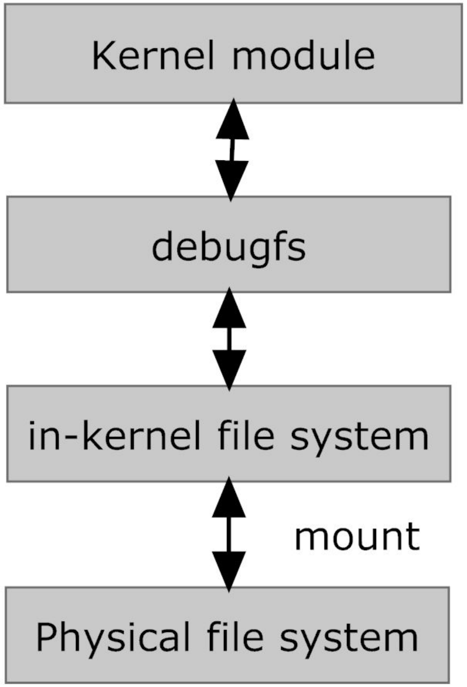


Monitor Mode



- The interface does not join to any network
- Used for passive sniffing.
- mac80211 sends upstream the unaltered IEEE 802.11 MAC
- radiotap includes physical layer information such as received channel, signal quality, signal to noise ratio, antenna and modulation scheme
- Sniffing tools such as Wireshark use pcap function to get these packets to the application layer.
- Packet injection
  - It is possible to inject random IEEE 802.11 MAC frames using the radiotap header and monitor mode WLAN network interface

debugfs



- in-kernel file-system
- Used for kernel development
- Used to examine and change the values of kernel module variables

Thank You

Questions ?

Publication

- Vipin M, Srikanth S. (2010), ‘Analysis of Open Source Drivers for IEEE 802.11 WLANs’ International Conference on Wireless Communication and Sensor Computing 2010. pp 66-70.

Reference Slides

Comparison of MADWiFi and ath9k

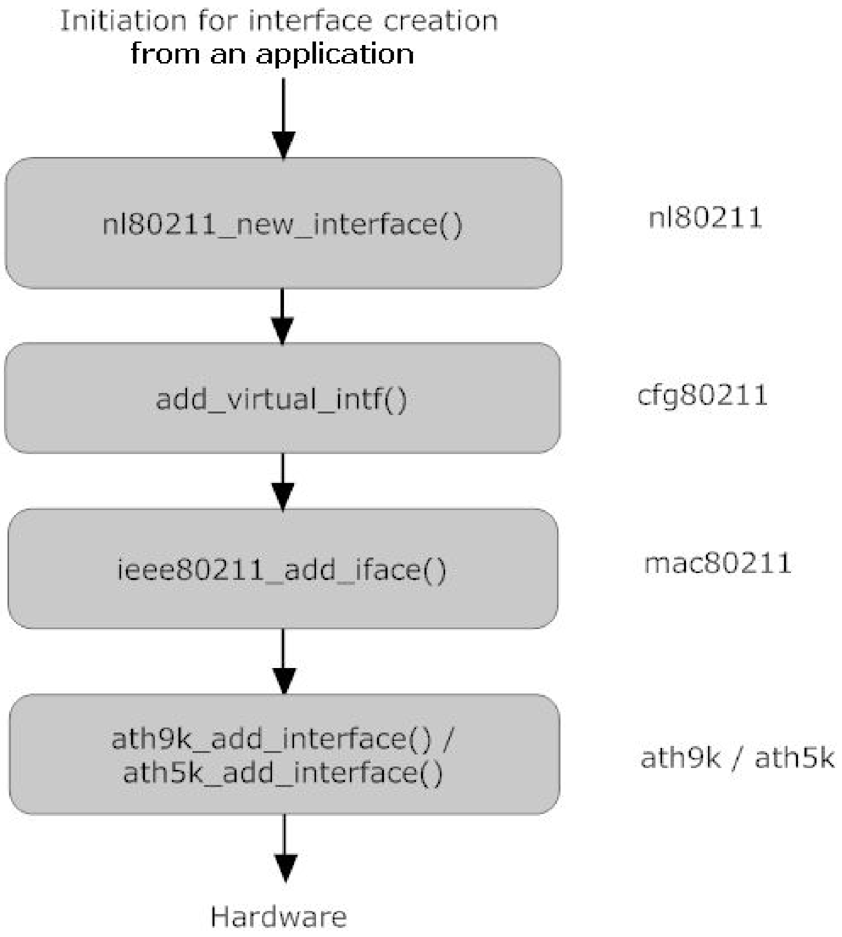
MADWiFi

- Use closed source HAL
  - Depend on HAL release
  - Even though openHAL is there it work with madwifi-old
- Used net80211 stack of BSD
  - Stack was modified to work with the driver.
- Support for a,b,g and e,i.
- Work with a variety of cards.
  - It support some of 11n cards in legacy mode.
- Support multiple modes
  - STA,Ad-Hoc,AP,Monitor

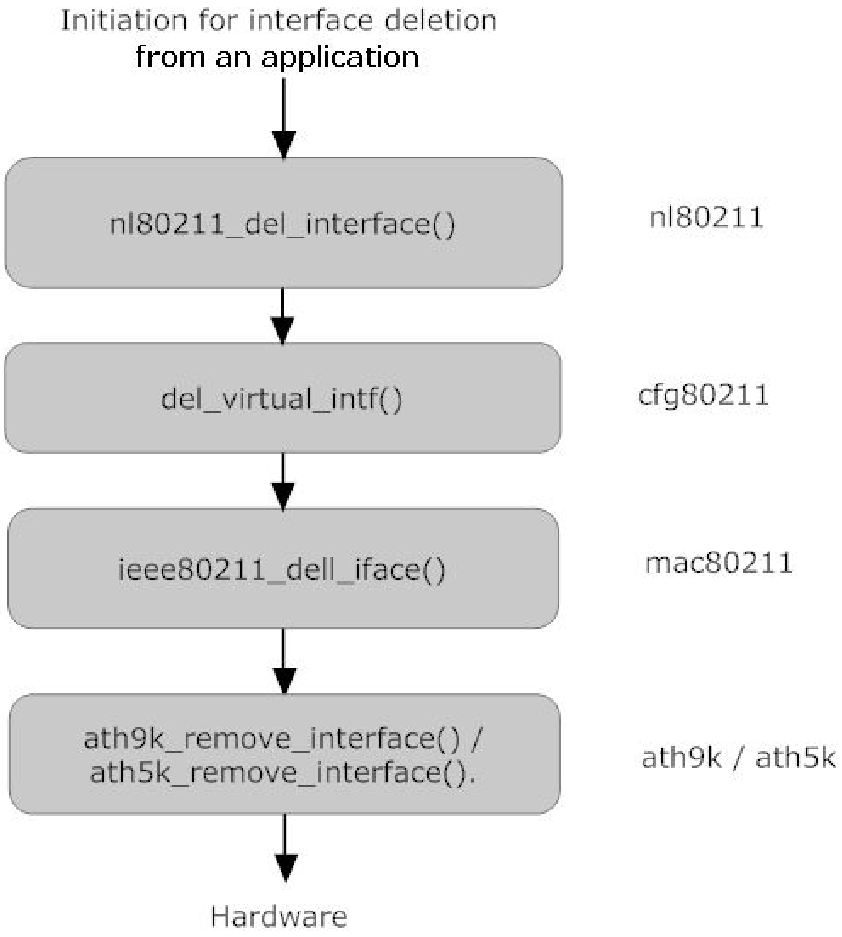
ath9k

- Fully open source.
  - Only depend on H/W
  - Atheros support and some reverse engineering all the futures are added.
- Use mac80211 stack
  - Mac80211 stack is shared with other drivers also.
- Support all e,i
  - As mac80211 is same it should be able work in a,b,g as the client support.
- Special driver for 11n devices.
  - This is specially for 11n cards.
- Support multiple modes
  - Suport STA,Monitor
  - Ad-Hoc , AP mode is not complete

Creating an interface



Delete an interface



Initiation for interface deletion from an application

nl80211_del_interface()

nl80211

del_virtual_intf()

cfg80211

ieee80211_dell_iface()

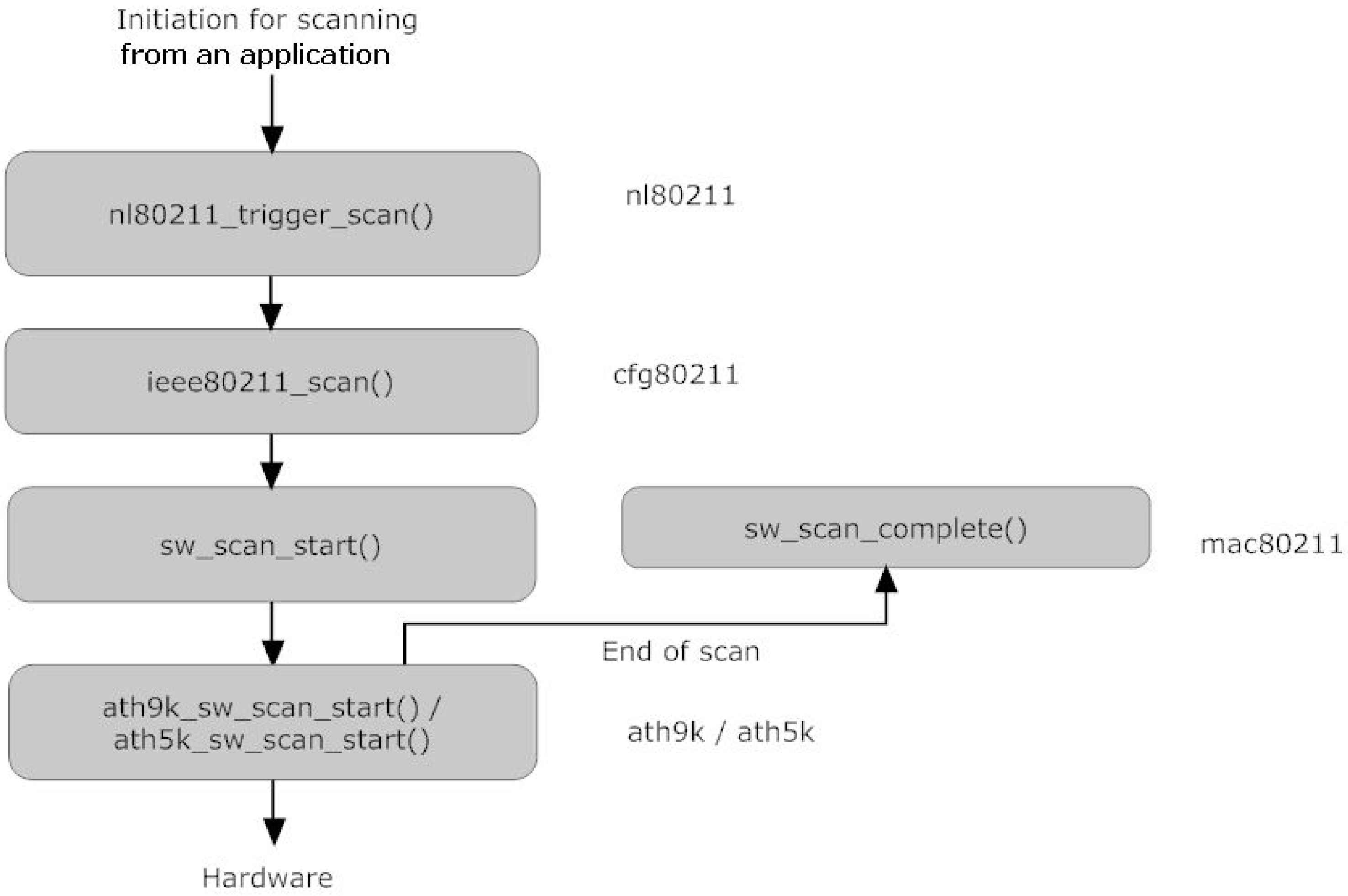
mac80211

ath9k_remove_interface() / ath5k_remove_interface().

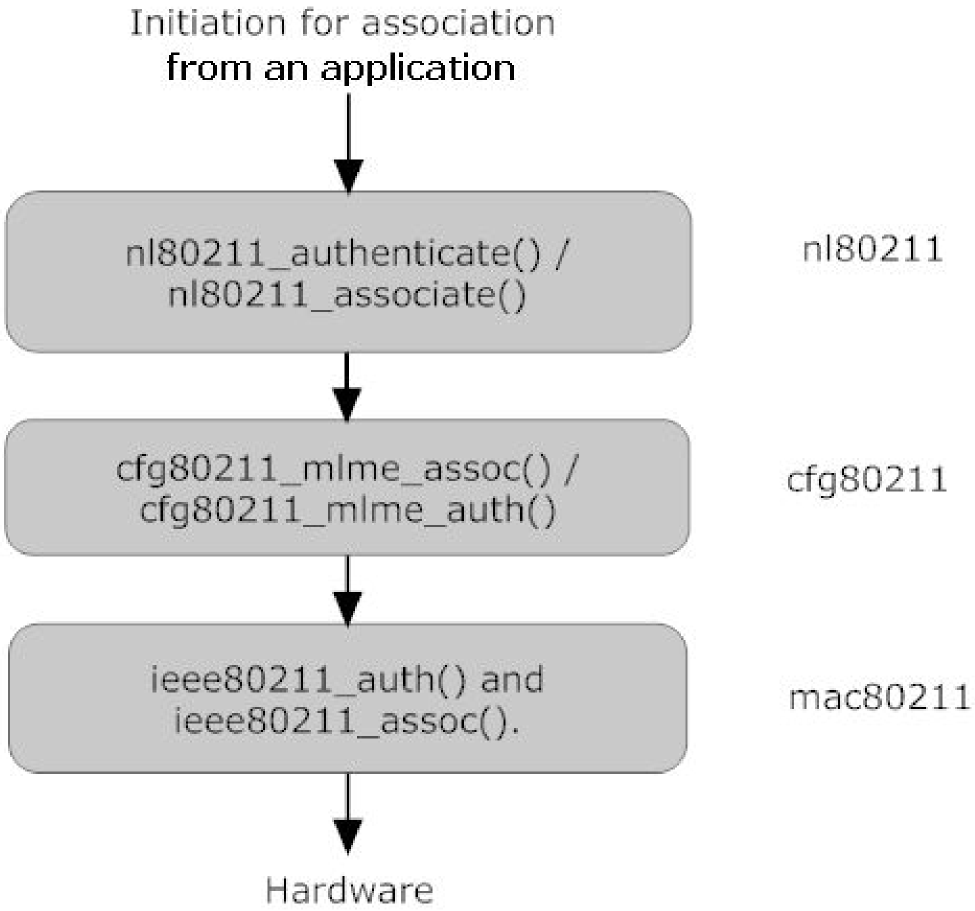
ath9k / ath5k

Hardware

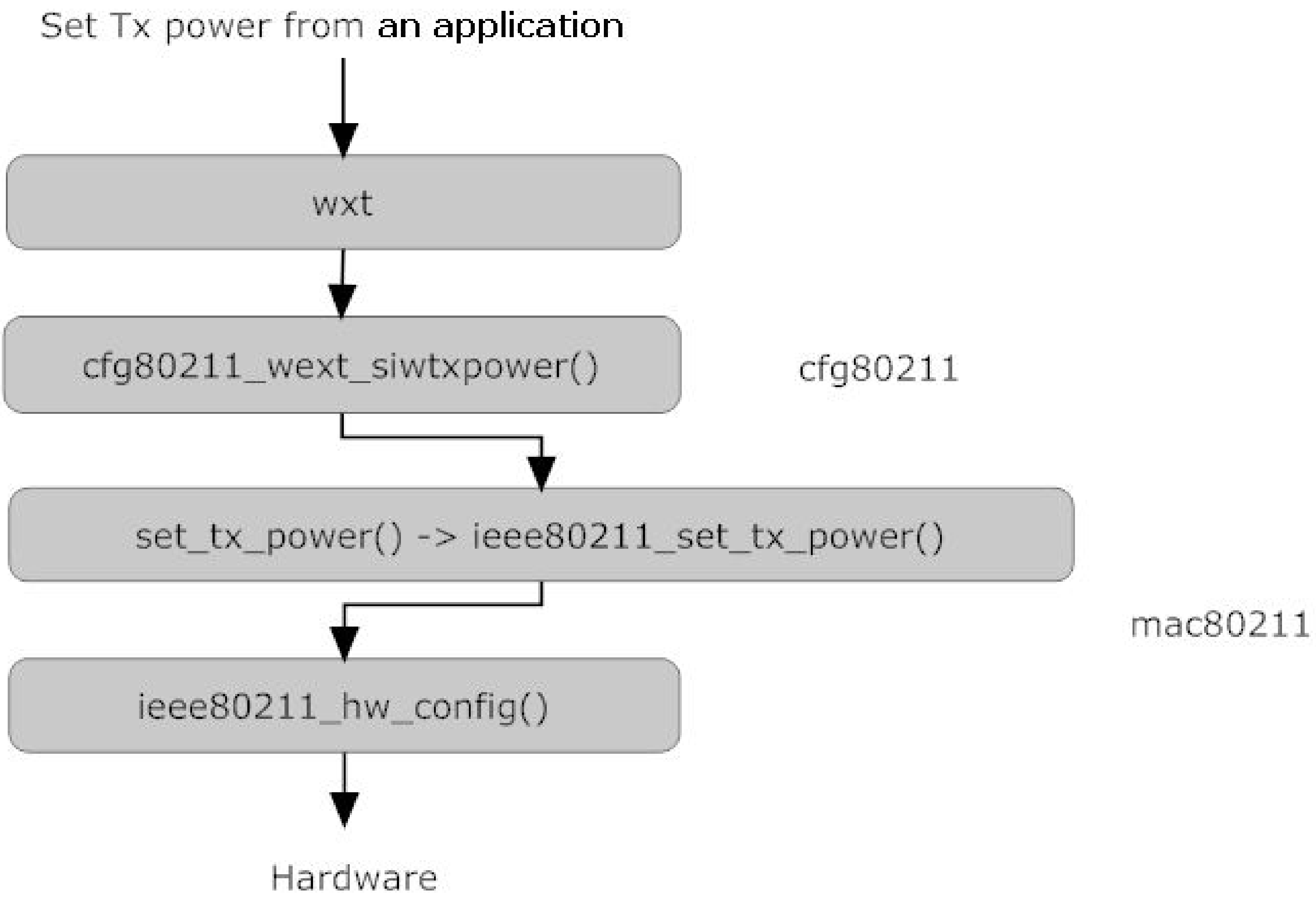
Scanning


Initiation for scanning
from an application
nl80211_trigger_scan()
nl80211
ieee80211_scan()
cfg80211
sw_scan_start()
sw_scan_complete()
mac80211
End of scan
ath9k_sw_scan_start() /
ath5k_sw_scan_start()
ath9k / ath5k
Hardware

Association



Tx power


Set Tx power from an application
wxt
cfg80211_wext_siwtxpower()
cfg80211
set_tx_power() -> ieee80211_set_tx_power()
mac80211
ieee80211_hw_config()
Hardware