WLAN overview

Contents

WLAN overview

Stable: 03.10.2019 - 12:51 / Revision: 03.10.2019 - 12:50

This article explains how the WLAN framework is composed, how to configure it, and how to use it.

Contents

1 Purpose

A wireless LAN (WLAN) is a wireless computer network that links two or more devices using wireless communication to form a local area network (LAN) within a limited area such as a home, school, computer laboratory, campus, office building etc. This gives to the users the ability to move around within the area and yet still be connected to the network. Through a gateway, a WLAN can also provide a connection to the wider Internet.

Linux® wireless subsystem contains two major blocks: cfg80211 and mac80211, and they help the WiFi driver to interface with rest of the kernel and user space.

In particular, cfg80211 provides configuration management services in the kernel. It also provides the management interface between the kernel and user space via nl80211.

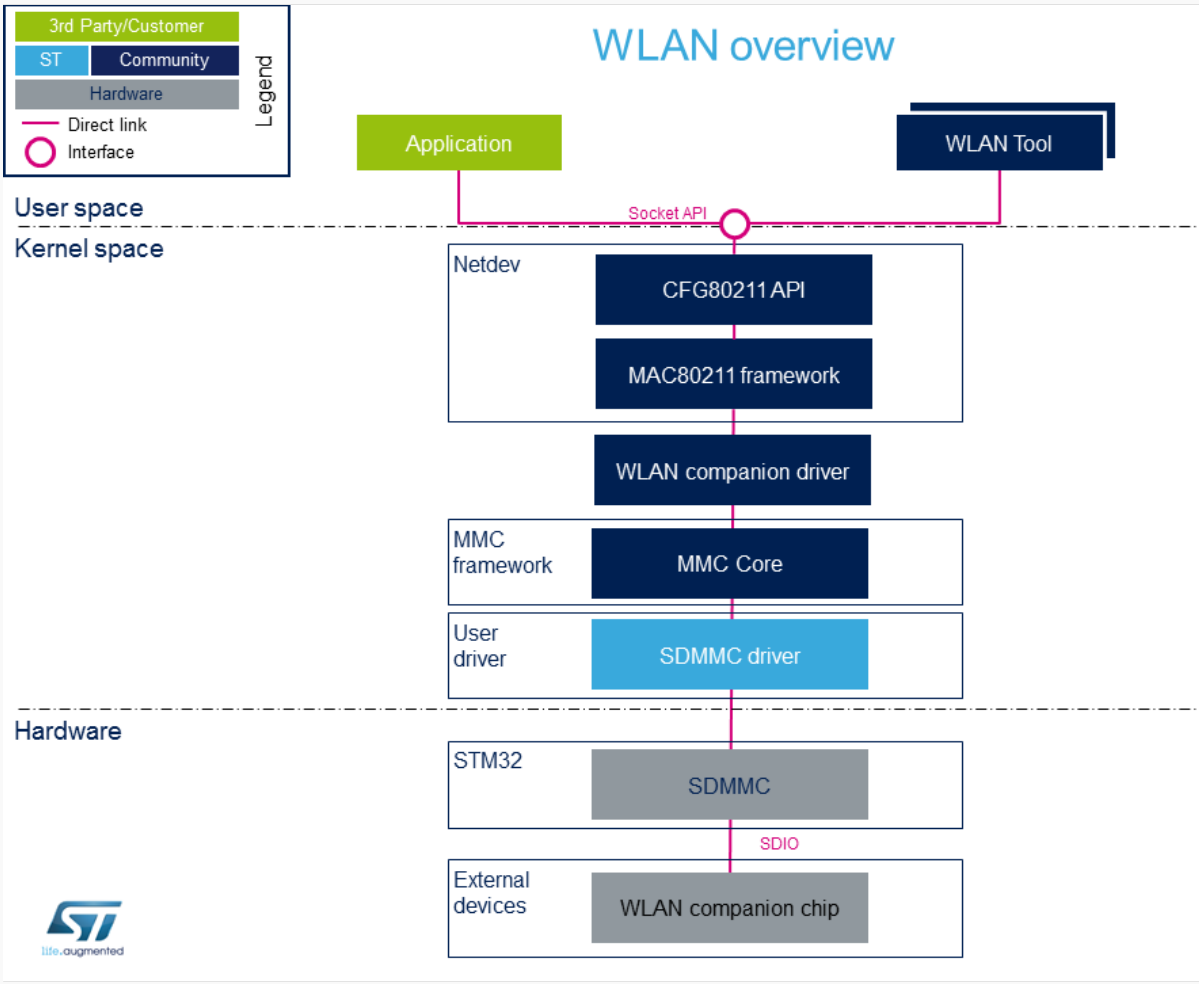
Both soft MAC and full MAC devices need to work with cfg80211. Mac80211 is a driver API that supports only software MAC devices

WLAN can be used in many different use cases, as mentioned in How to use WLAN section:

- How to configure a wlan interface on client mode: How to configure a wlan interface on client mode

- How to configure a wlan interface on hotspot mode: How to configure a wlan interface on hotspot mode

2 System overview



WLAN Overview

2.1 Description of the components

From User space to hardware

- **Application** (User space)

There are lots of application which are using WLAN: such as Internet Browser, Streaming applications, FTP applications.

- **WLAN tool** (User space)

There are a set of utilities to manage wlan networks: Network tools

- **CFG80211 API** (Kernel space)

cfg80211[1] is the configuration API for IEEE 802.11[2] devices in Linux.
- It bridges the User space and drivers, and offers some utility functionality associated with IEEE 802.11.
- It must be used, directly or indirectly via mac80211, by all modern wireless drivers in Linux, so that they offer a consistent API via nl80211[3].
- It is interfaced with Netlink[4] socket.

- **MAC80211 framework** (Kernel space)

MAC80211 is a subsystem to the Linux kernel, which implements shared code for soft-MAC/half-MAC wireless devices[5]

- **WLAN companion driver** (Kernel space)

WLAN companion driver register and control WLAN device.

- **MMC framework: MMC Core** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[6] / secure digital (**SD**)[7] / secure digital input/output (**SDIO**)[8].

The communication link between MP1 and WLAN device is the SDIO bus.

- **SDMMC driver** (Kernel space)

More information in MMC overview

- **STM32: SDMMC** (Hardware)

More information in MMC overview

- **External devices** (Hardware)

WLAN companion chip

2.2 APIs description

MAC80211 is new wireless driver API, which implements the shared code for soft-MAC/half-MAC wireless devices[9]

cfg80211[10] is the new driver configuration API for IEEE 802.11[11] devices in Linux.

3 Configuration

The WLAN API is not activated by default in ST deliveries. To active it, you can use Linux Menuconfig tool: Menuconfig or how to configure kernel and select:

For Network features:

```
[*] Networking support  --->
    [*] Networking options  --->
        [*] Packet socket
        [*] TCP/IP networking
          [*] IP: kernel level autoconfiguration
            [*] IP: DHCP support
            [*] IP: BOOTP support
            [*] IP: RARP support
        [*] INET: socket monitoring interface
```

```
        [*] The IPv6 protocol
        [*] DNS Resolver support
    [*]   Wireless  --->
        [*]   cfg80211 - wireless configuration API
                [*]   cfg80211 wireless extensions compatibility
        [*]   Generic IEEE 802.11 Networking Stack (mac80211)
```

For example if the companion chip is Murata chip 1DX[12]

```
[*] Device Drivers  --->
    [*]   Network device support  --->
        [*]   Wireless LAN  --->
            [*]   Broadcom devices
                [*]   Broadcom FullMAC WLAN driver
```

For STM32 SDMMC : see SDMMC configuration

3.1 Device tree

The DT bindings documentation deals with all required or optional device tree properties.

Detailed DT configuration for STM32 peripherals: WLAN device tree configuration.

4 How to use WLAN

4.1 How to use the WLAN user space interface

Please see examples based on the following use cases:

- How to setup wifi connection: How to setup wifi connection
- How to configure a wlan interface on client mode: How to configure a wlan interface on client mode
- How to configure a wlan interface on hotspot mode: How to configure a wlan interface on hotspot mode

5 How to trace and debug the framework

5.1 How to monitor

Please give instruction to help the developer/user to get information about this framework in the Linux file system

5.1.1 How to watch link quality

Proc filesystem provides information about Quality link:

```
Board $> cat /proc/net/wireless
Inter-| sta-|   Quality        |   Discarded packets               | Missed | WE
face | tus | link level noise |  nwid  crypt   frag  retry   misc | beacon | 22
wlan0: 0000    0     0     0        0      0      0      0      0        0
```

Example of stat:

```
Board $> watch -d -n 3 "iw dev wlan0 station dump; iwconfig wlan0; cat /proc/net
/wireless"
```

```
Every 3.0s: iw dev wlan0 station dump; iwconfig wlan0; cat /proc/net/wir...  stm32mp1:
Sun Nov  4 16:32:52 2018

Station 00:16:b6:2c:47:36 (on wlan0)
       inactive time:  0 ms
       rx bytes:       11001
       rx packets:     37
       tx bytes:       13077
       tx packets:     83
       tx failed:      0
       signal:         -72 [-72] dBm
       tx bitrate:     12.0 MBit/s
       rx bitrate:     1.0 MBit/s
       authorized:     yes
       authenticated:  yes
       associated:     yes
       WMM/WME:        no
       TDLS peer:      yes
       DTIM period:    1
       beacon interval:100
       short preamble: yes
       short slot time:yes
       connected time: 55 seconds
wlan0     IEEE 802.11  ESSID:"NETWORK1"
          Mode:Managed  Frequency:2.462 GHz  Access Point: 00:16:B6:2C:47:36
          Bit Rate=12 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=38/70  Signal level=-72 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

Inter-| sta-|   Quality        |   Discarded packets               | Missed | WE
face | tus | link level noise |  nwid  crypt   frag  retry   misc | beacon | 22
wlan0: 0000   38.  -72.  -256        0      0      0      0      0        0
```

5.2 How to trace

This part is as example in case the companion chip is Murata chip

5.2.1 How to verify than the WLAN driver is well probed

- In dmesg log, check "brcmfmac" logs :

```
[   67.306154] brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Aug  6 2017 23:
19:25 version 7.45.98.30 (r666241 CY) FWID 01-f0b000
[   67.326146] brcmfmac: brcmf_c_preinit_dcmds: CLM version = API: 12.2 Data: 7.11.15
Compiler: 1.24.2 ClmImport: 1.24.1 Creation: 2014-05
[   67.676323] brcmfmac: brcmf_cfg80211_reg_notifier: not a ISO3166 code (0x30 0x30)
```

5.2.2 How to debug the WLAN driver

5.2.2.1 Add dynamic debug firmware traces

Need to activate in the kernel config: CONFIG_DYNAMIC_DEBUG, more info on the dynamic debug How_to_use_the_kernel_dynamic_debug

```
# cd /sys/kernel/debug/dynamic_debug/
```

Check all functions used to manage firmware:

```
# cat control | grep firmware
drivers/base/firmware_class.c:339 [firmware_class]__fw_free_buf =_ "%s: fw-%s buf=%p
data=%p size=%u\012"
drivers/base/firmware_class.c:462 [firmware_class]fw_set_page_data =_ "%s: fw-%s buf=%
p data=%p size=%u\012"
drivers/base/firmware_class.c:1102 [firmware_class]_request_firmware_prepare =_ "using
built-in %s\012"
drivers/base/firmware_class.c:290 [firmware_class]__allocate_fw_buf =_ "%s: fw-%s buf=%
p\012"
drivers/base/firmware_class.c:1194 [firmware_class]_request_firmware =_ "firmware: %s
loading timed out\012"
drivers/base/firmware_class.c:423 [firmware_class]fw_get_filesystem_firmware =_
"loading %s failed with error %d\012"
drivers/base/firmware_class.c:429 [firmware_class]fw_get_filesystem_firmware =_
"direct-loading %s\012"
```

Add print info "+p" in all firmware functions (p: causes a printk() message to be emitted to dmesg)

```
# echo "file drivers/base/firmware_class.c +p" > control
```

Now "p" option is added in all firmware functions.

```
# cat control | grep firmware
drivers/base/firmware_class.c:339 [firmware_class]__fw_free_buf =p "%s: fw-%s buf=%p
data=%p size=%u\012"
drivers/base/firmware_class.c:462 [firmware_class]fw_set_page_data =p "%s: fw-%s buf=%
p data=%p size=%u\012"
drivers/base/firmware_class.c:1102 [firmware_class]_request_firmware_prepare =p "using
built-in %s\012"
drivers/base/firmware_class.c:290 [firmware_class]__allocate_fw_buf =p "%s: fw-%s buf=%
p\012"
drivers/base/firmware_class.c:1194 [firmware_class]_request_firmware =p "firmware: %s
loading timed out\012"
drivers/base/firmware_class.c:423 [firmware_class]fw_get_filesystem_firmware =p
"loading %s failed with error %d\012"
drivers/base/firmware_class.c:429 [firmware_class]fw_get_filesystem_firmware =p
"direct-loading %s\012"
```

5.2.2.2 FMAC debug

5.2.2.2.1 Enable debug features in the defconfig file

- Enable CPTCFG_BRCMDBG and CONFIG_DEBUG_FS
- Rebuild your kernel

5.2.2.2.2 Enable brcmfmac debug log

- Message levels are listed in **drivers/net/wireless/broadcom/brcm80211/brcmfmac/debug.h**

```
/* message levels */
#define BRCMF_TRACE_VAL 0x00000002
#define BRCMF_INFO_VAL 0x00000004
#define BRCMF_DATA_VAL 0x00000008
#define BRCMF_CTL_VAL 0x00000010
#define BRCMF_TIMER_VAL 0x00000020
#define BRCMF_HDRS_VAL 0x00000040
#define BRCMF_BYTES_VAL 0x00000080
#define BRCMF_INTR_VAL 0x00000100
#define BRCMF_GLOM_VAL 0x00000200
#define BRCMF_EVENT_VAL 0x00000400
#define BRCMF_BTA_VAL 0x00000800
#define BRCMF_FIL_VAL 0x00001000
#define BRCMF_USB_VAL 0x00002000
#define BRCMF_SCAN_VAL 0x00004000
#define BRCMF_CONN_VAL 0x00008000
#define BRCMF_BCDC_VAL 0x00010000
#define BRCMF_SDIO_VAL 0x00020000
#define BRCMF_MSGBUF_VAL 0x00040000
#define BRCMF_PCIE_VAL 0x00080000
#define BRCMF_FWCON_VAL 0x00100000
```

```
$ modprobe brcmfmac debug=${BRCMF_Message_Level}
$ dmesg -n 8
```

5.2.2.2.3 Examples

- Add TRACE and INFO

```
$ modprobe brcmfmac debug=0x6
```

- Add TRACE, INFO and SDIO

```
$ modprobe brcmfmac debug=0x20006
```

- Add TRACE, INFO and WIFI_FW_LOG

```
$ modprobe brcmfmac debug=0x00100006
```

5.2.2.2.4 How to check wlreg_on status and voltage setting

```
$ cat /sys/kernel/debug/regulator/regulator_summary
```

6 Source code location

The source files are located inside the Linux kernel.

- **Broadcom wlan driver**: of.c[13]

7 References

- [1], cfg80211
- [2], IEEE_802.11
- [3], nl80211
- [4], Netlink
- [5], mac80211
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification
- [6], mac80211
- [7], cfg80211
- [8], IEEE_802.11
- [9], 1DX
- [https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/net/wireless/broadcom/brcm80211/brcmfmac/of.c],of.c

Application programming interface

MultimediaCard

Secure digital

Secure digital input/output

Device Tree

How to configure a wlan interface on client mode

Stable: 03.02.2020 - 08:43 / Revision: 03.02.2020 - 08:31

Invalid target: no reviewed revision corresponds to the given ID.

Return to How to configure a wlan interface on client mode.

How to configure a wlan interface on hotspot mode

Stable: 03.02.2020 - 08:43 / Revision: 03.02.2020 - 08:30

Invalid target: no reviewed revision corresponds to the given ID.

Return to How to configure a wlan interface on hotspot mode.

Network tools

Stable: 04.02.2020 - 07:47 / Revision: 04.02.2020 - 07:37

Invalid target: no reviewed revision corresponds to the given ID.

Return to Network tools.

MMC overview

Stable: 16.09.2019 - 15:05 / Revision: 16.09.2019 - 15:04

Invalid target: no reviewed revision corresponds to the given ID.

Return to MMC overview.

Menuconfig or how to configure kernel

Stable: 31.01.2020 - 12:57 / Revision: 31.01.2020 - 12:52

Invalid target: no reviewed revision corresponds to the given ID.

Return to Menuconfig or how to configure kernel.

Device tree

Stable: 04.02.2020 - 07:47 / Revision: 04.02.2020 - 07:34

Invalid target: no reviewed revision corresponds to the given ID.

Return to Device tree.

WLAN device tree configuration

Stable: 06.02.2020 - 15:08 / Revision: 06.02.2020 - 15:07

Invalid target: no reviewed revision corresponds to the given ID.

Return to WLAN device tree configuration.

How to setup wifi connection

Stable: 09.03.2020 - 08:21 / Revision: 06.03.2020 - 13:51

Invalid target: no reviewed revision corresponds to the given ID.

Return to How to setup wifi connection.

How to use the kernel dynamic debug

Stable: 26.09.2019 - 08:34 / Revision: 26.09.2019 - 08:32

Invalid target: no reviewed revision corresponds to the given ID.

Return to How to use the kernel dynamic debug.